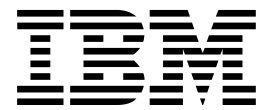


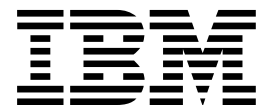
DFSORT



Getting Started with DFSORT

Release 14

DFSORT



Getting Started with DFSORT

Release 14

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page vii.

Ninth Edition (September 1998)

This edition replaces and makes obsolete the previous edition, SC26-4109-07. The technical changes for this edition are summarized under "Summary of Changes," and are indicated by a vertical bar to the left of a change.

This edition applies to Release 14 of DFSORT (5740-SM1) and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
RCF Processing Department
G26/M86 050
5600 Cottle Road
SAN JOSE, CA 95193-0001
U.S.A.

Or, you can send us comments about this book electronically:

- IBMLink from US and IBM Network: STARPUBS at SJEVM5
- IBMLink from Canada: STARPUBS at TORIBM
- IBM Mail Exchange: USIB3VVD at IBMMAIL
- Internet: starpubs@sjevm5.vnet.ibm.com or, starpubs at sjevm5.vnet.ibm.com
- Fax (US): 1-800-426-6209

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1983, 1998. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Programming Interface Information	vii
Trademarks	vii
Preface	ix
About This Book	ix
DFSORT Publications	ix
DFSORT Library Softcopy Information	x
Related Publications	x
Referenced Publications	x
Summary of Changes	xiii
Ninth Edition, September 1998	xiii
New Programming Support for Release 14	xiii
New Programming Support for Release 13 (PTFs after April, 1996)	xvi
New Programming Support for Release 13 (PTFs - April, 1996)	xvi

Part 1. Introduction 1

Chapter 1. What is DFSORT?	3
DFSORT on the World Wide Web	3
DFSORT FTP Site	3
Sorting Data Sets	3
Merging Data Sets	4
Copying Data Sets	5
What Else Can You Do with DFSORT?	5
Creating and Running DFSORT Jobs	5
Writing Jobs	6
Creating Jobs Interactively	6
Running Jobs	6
Using the Sample Bookstore Data Sets	6
Creating Your Sample Input and Output Data Sets	8

Part 2. Learning to Write JCL and DFSORT Control Statements 9

Chapter 2. Sorting, Merging, and Copying Data Sets	11
Sorting Data Sets	11
Sorting by Multiple Fields	13
Continuing a Statement	14
Sorting Data Sets with the JCL EXEC Statement	15
Merging Data Sets	16
Writing the MERGE Control Statement	17
Merging Data Sets with the JCL EXEC Statement	18
Copying Data Sets	19
Specifying COPY on the SORT, MERGE, or OPTION Statement	19
Copying Data Sets with the JCL EXEC Statement	20
Chapter 3. Tailoring the Input Data Set with INCLUDE and OMIT	21
Writing the INCLUDE Statement	21

Writing the OMIT Statement	24
Allowable Comparisons for INCLUDE and OMIT	25
Writing Constants	26
Chapter 4. Summing Records	27
Writing the SUM Statement	27
Suppressing Records with Duplicate Control Fields	29
Handling Overflow	29
Chapter 5. Reformatting Records	31
Reformatting Records After Sorting	31
Reordering Fields to Reserve Space	32
Inserting Binary Zeros	33
Inserting Blanks	33
Inserting Constants	34
Character Strings	35
Hexadecimal Strings	35
Setting Up the Report Format	35
Reformatting Records Before Sorting	37
Using Other Statements with INREC	38
Preventing Overflow When Summing Values	40
Chapter 6. Creating Multiple Output Data Sets and Reports	41
Creating Multiple Copies	42
Creating Multiple Output Data Sets with Unique Content	43
Creating Reports: ICETOOL vs OUTFIL	45
Using OUTFIL to Create Reports	45
Using the FNames and Lines Parameters	49
Using the HEADER2 Parameter	50
Using the OUTREC Parameter	52
Using the TRAILER1 Parameter	54
Chapter 7. Calling DFSORT from a Program	57
Passing Control Statements	57
Calling DFSORT from a COBOL Program	57
Sorting Records	57
Merging Records	60
Sorting with COBOL FASTSORT	63
Calling DFSORT from a PL/I Program	63
Chapter 8. Overriding Installation Defaults	65
Specifying PARM Parameters on a JCL EXEC Statement	65
Writing an OPTION Control Statement	65
Specifying DFSPARM Parameters	66
Chapter 9. Using DFSORT Efficiently	67
Be Generous with Main Storage	67
Use High-Speed DASD or Hiperspace	67
Eliminate Unnecessary Fields with INREC	68
Eliminate Unnecessary Records with INCLUDE or OMIT	68
Reduce File Size with STOPAFT and SKIPREC	68
Consolidate Records with SUM	68
Create Multiple Output Data Sets with OUTFIL	69
Run DFSORT with JCL	69

Use FASTSRT with COBOL	69
Avoid Options That Might Degrade Performance	69

Part 3. Learning to Use ICETOOL	71
--	-----------

Chapter 10. Using the ICETOOL Utility	73
ICETOOL Operators	73
Input Data Sets	74
Creating an ICETOOL Job	75
Writing Required JCL Statements	75
ICETOOL Comment and Blank Statements	76
Printing Statistics For Numeric Fields	77
Continuing an Operator Statement	78
Statistics For Record Lengths	79
Creating Identical Sorted Data Sets	79
Creating Different Subsets of a Sorted Data Set	82
Creating Multiple Unsorted Data Sets	84
Counting Values in a Range	85
Printing Simple Reports	87
Printing Tailored Reports	88
Using Formatting Items	90
Edit Masks	90
Division	92
Leading, Floating and Trailing Characters	92
Printing Sectioned Reports	93
Printing How Many Times Fields Occur	96
Selecting Records by Field Occurrences	97
Complete ICETOOL Job and TOOLMSG Output	98

Part 4. Learning to Use Symbols	103
--	------------

Chapter 11. Defining and Using Symbols	105
Creating the SYMNAMES Data Set	105
Defining Symbols for Fields	105
Using Symbols for Fields in DFSORT Statements	107
Using Symbols for Fields in ICETOOL Operators	108
Defining and Using Symbols for Constants	109

Part 5. Appendixes	113
-------------------------------------	------------

Appendix A. Using the DFSORT Sample Data Sets	115
Appendix B. The Sample Bookstore Data Sets	117
Appendix C. Processing Order of Control Statements	121
Summary of Changes	123
Release 13	123
New Programming Support for Release 13	123
New Programming Support for Release 12 (PTFs)	125
New Device Support for Release 12 (PTFs)	125

Index	127
------------------------	-----

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as a Programming Interface of DFSORT.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of DFSORT. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

Programming Interface information

End of Programming Interface information

Trademarks

The following terms are trademarks or service marks of the IBM Corporation in the United States or other countries or both:

DFSMS/MVS	IBM
DFSORT	MVS/ESA
Hipersorting	OS/390
Hiperspace	RACF

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of other companies.

Preface

Getting Started with DFSORT is a user's book and tutorial for DFSORT (Data Facility Sort). You should read it if you are new to DFSORT Licensed Program 5740-SM1 and need to learn the basics of using DFSORT to process data sets. Experienced DFSORT users can use this book as a general guide to DFSORT.

The chapters in this book assume that you have used job control language (JCL) and understand how to work with data sets. You should also know what data sets are available at your site.

About This Book

This book gives you all of the information and instructions you need to build and submit DFSORT jobs. You can use DFSORT by writing JCL and DFSORT program control statements, or by entering values on interactive, full-screen panels. This book describes the first method of creating DFSORT jobs. The second method is detailed in *DFSORT Panels Guide*.

New users should work through *Getting Started* from cover to cover. Each task explained in this book builds on knowledge gained in previous tasks. The Table of Contents lists the main tasks, and summaries are included in the chapters. If you have previous experience with these tasks, you can proceed from here directly to the tutorials that begin in Chapter 2, "Sorting, Merging, and Copying Data Sets" on page 11.

Chapter 1, What is DFSORT? is an overview of the basic principles of sorting, merging, and copying, and explains how to create and use the sample bookstore data sets for the examples in this book.

Chapters 2-9 show you how to create and process DFSORT jobs by writing JCL and DFSORT program control statements to sort, merge, and copy data sets. These chapters also detail DFSORT's methods for arranging data sets and generating reports.

Chapter 10 shows you how to create and process ICETOOL jobs by writing JCL and ICETOOL statements. ICETOOL is a multipurpose DFSORT utility that uses the capabilities of DFSORT to perform multiple operations on one or more data sets in a single step.

Chapter 11 shows you how to define symbols for fields and constants and use them in your DFSORT control statements and ICETOOL operators.

Several appendixes and an index follow the chapters.

DFSORT Publications

Getting Started with DFSORT is a part of a more extensive DFSORT library. The additional books in the library are listed below. You can order a complete set of DFSORT publications with the order number SBOF-1243, except for *DFSORT Licensed Program Specifications* (GC33-4032), which must be ordered separately.

Task	Publication Title	Order Number
Application Programming	<i>DFSORT Application Programming Guide Release 14</i>	SC33-4035
Diagnosing Failures and Interpreting Messages	<i>DFSORT Messages, Codes and Diagnosis Guide Release 14</i>	SC26-7050
Planning for and Customizing DFSORT	<i>DFSORT Installation and Customization Release 14</i>	SC33-4034
Quick Reference	<i>DFSORT Reference Summary Release 14</i>	SX33-8001
Tuning DFSORT	<i>DFSORT Tuning Guide Release 14</i>	SC26-3111
Learning to Use DFSORT Panels	<i>DFSORT Panels Guide</i>	GC26-7037

DFSORT Library Softcopy Information

The DFSORT library is available on CD-ROM.

Order Number	Title
SK2T-0710	<i>IBM Online Library Omnibus Edition MVS Collection</i>
SK2T-6700	<i>IBM Online Library Omnibus Edition OS/390 Collection</i>

Related Publications

For up-to-date descriptions of all of the books that support OS/390, refer to the *OS/390 Information Roadmap*, GC28-1727.

In the course of learning how to use DFSORT, you might also want to refer to the publications listed in the table below.

Short Title	Publication	Order Number
JCL Reference	<i>MVS/ESA JCL Reference</i> (for MVS/ESA SP Version 5)	GC28-1479
	<i>MVS/ESA JCL Reference</i> (for MVS/ESA SP Version 4)	GC28-1654
JCL User's Guide	<i>MVS/ESA JCL User's Guide</i> (for MVS/ESA SP Version 5)	GC28-1473
	<i>MVS/ESA JCL User's Guide</i> (for MVS/ESA SP Version 4)	GC28-1653

Referenced Publications

Within the text of this document, references are made to the following publications:

Short Title	Publication	Order Number
Application Programming Guide	<i>DFSORT Application Programming Guide Release 14</i>	SC33-4035
Panels Guide	<i>DFSORT Panels Guide</i>	GC26-7037
Messages, Codes and Diagnosis	<i>DFSORT Messages, Codes and Diagnosis Guide Release 14</i>	SC26-7050

A more comprehensive list of related publications appears in *DFSORT Application Programming Guide*.

For more information on using DFSORT with COBOL or PL/I, see the Programmer's Guide describing the compiler version available at your site.

Summary of Changes

Ninth Edition, September 1998

New Programming Support for Release 14

Symbols for Fields and Constants

DFSORT now provides a simple and flexible method for using symbols in DFSORT and ICETOOL statements. You can define and use a symbol for any field or constant that is recognized in a DFSORT control statement or ICETOOL operator. This makes it easy to create and reuse collections of symbols (that is, mappings) for your frequently used data.

In addition, you can obtain and use collections of DFSORT symbols created specifically for data associated with other products (for example, RACF, DFSMSrmm and DCOLLECT) or by your site.

DFSORT symbols can increase your productivity by automatically providing the positions, lengths and formats of the fields and the literals, numbers, and bit flags of the constants, associated with the particular records you are processing with DFSORT or ICETOOL.

Improvements in Performance, Capacity and Storage Usage

Blockset copy and merge applications can now use storage above 16MB virtual, providing improved performance and virtual storage constraint relief.

Blockset copy and merge modules will now reside above 16MB virtual, providing virtual storage constraint relief.

DFSORT can now handle a significantly larger number of INCLUDE and OMIT conditions.

DFSORT can now handle a significantly larger number of SUM fields.

The upper limit for the number of JCL and dynamically allocated work data sets that can be specified and used by DFSORT's Blockset technique has been raised from 100 to 255. The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application. Any valid ddname of the form SORTWKdd or SORTWKd can now be used for DASD work data sets (for example, SORTWK01, SORTWK3, SORTWK2, SORTWK#5, SORTWKA, SORTWKXY and so on).

The upper limit for the number of input data sets that can be specified and used for a Blockset merge application has been raised from 16 to 100. The use of more merge input data sets increases the maximum amount of data DFSORT can process in a single merge application.

Time-of-Day Option Controls

New time-of-day installation modules (ICETD1-4) allow different sets of installation defaults to be used, based on the day and time DFSORT applications run. Each environment installation module (ICEAM1-4) can enable one or more time-of-day installation modules. This capability allows new levels of control for installation defaults. For example, larger storage, hiperspace and data space limits could be used only for batch program-invoked DFSORT applications that run off-shift during the week, and all weekend.

Repackaging

The product has been repackaged to simplify installation and customization:

- IBM's DFSORT, DFSMSdfp, and MVS/DFP teams have simplified the process of replacing IEBGENER with ICEGENER. You now only need to apply a DFSMS or DFP PTF that supplies an alias of "IEBGENR" for IEBGENER and place ICEGENER with an alias of "IEBGENER" ahead of IEBGENER in the system's search order for programs.
- The number of FMIDs has been reduced from 10 to 3.
- The number of libraries required to install DFSORT has been reduced from 40 to 26.
- DFSORT R14 now supports a single installation of the product for both resident and nonresident features. This allows you to decide how to use DFSORT independent of the installation method, thus reducing the number of decisions you have to make at installation time.
- All FMIDs in DFSORT R14 can be installed together, including the FMIDs for both English and Japanese messages and panels.

OUTFIL Processing Enhancements

OUTFIL now supports creation of multiple output records using the fields of the input record. This allows you to split each record into pieces, include a field in more than one record, include different fields in different records, and more.

OUTFIL now supports processing of variable-length input records which are too short to contain all specified OUTFIL OUTREC fields. OUTFIL's new VLFILL=byte operand can be used to replace missing bytes in OUTFIL OUTREC fields with the specified fill byte so the filled fields can be processed.

ICETOOL Enhancement

A new DISCARD(savedd) operand of ICETOOL's SELECT operator allows you to save the records that are not selected, in the savedd data set. Thus, in one pass, you can create an outdd data set with the records that meet your specified criteria, and a savedd data set with the records that do not meet your specified criteria. DISCARD(savedd) can be used to save the records discarded by ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v), FIRST or LAST.

Installation and Run-Time Option Enhancements

A new p% value for the EXPRES, EXPOLD, and EXPMAX installation options and the HIPRMAX installation and run-time options is now available. p% can be used to vary the limit DFSORT calculates for the corresponding option as a percentage of the configured expanded storage on the system at run time. If the configured expanded storage on a system changes, p% will cause a corresponding change in the run-time limit calculated for the corresponding option. When sharing DFSORT

installation options between systems, such as in a sysplex, p% can be used to tailor the limit DFSORT calculates for the corresponding option to the system on which the application runs.

A new SPANINC installation and run-time option allows you to specify what you want DFSORT to do if it detects incomplete spanned records. This gives you control over the action (continue by eliminating incomplete spanned records and recovering valid records, or terminate), type of message (informational or error) and return code (0, 4 or 16) for incomplete spanned records.

A new OVFO installation and run-time option allows you to specify what you want DFSORT to do when BI, FI, PD or ZD summary fields overflow. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for summary overflow.

A new PAD installation and run-time option allows you to specify what you want DFSORT to do when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for LRECL padding.

A new TRUNC installation and run-time option allows you to specify what you want DFSORT to do when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for LRECL truncation.

The IBM-supplied default for ICEMAC option DSA has been changed from 16MB to 32MB.

The IBM-supplied default for ICEMAC option GENER has been changed from IEBGENER to IEBGENR.

The maximum value for ICEMAC option OVERRGN has been changed from 64KB to 16128KB.

Other Enhancements

New messages ICE178I and ICE179A provide information about reallocation of VIO work data sets.

The option-in-effect messages (ICE127I-ICE133I) are now printed for Blockset copy and merge applications.

The user exit address constant can now be passed to E32 user exits for Blockset merge applications.

Null segments in variable spanned input records are now processed by DFSORT and no longer result in termination. A null segment means that there are no more segments in the block.

OS/390 and MVS/ESA Only

DFSORT Release 14 only supports the OS/390 and MVS/ESA environments. MVS/XA and VIRTDSP processing for MVS/XA are no longer supported.

New Programming Support for Release 13 (PTFs after April, 1996)

Additional Year 2000 Features

A new Y2S format can order and transform two-digit character or zoned decimal year data according to the century window, while handling binary zeros, blanks and binary ones in the year field as special indicators.

A new Y2B format can order and transform two-digit binary year data according to the century window.

FREE=CLOSE support for DFSPARM makes it possible to override the SORT statements generated by multiple COBOL SORT verbs in the same COBOL program.

OS/390 Registration

With OS/390 R2 and above, a check is performed to ensure that the DFSORT product is licensed for use, either as a feature of OS/390 or as a separate program product.

New Programming Support for Release 13 (PTFs - April, 1996)

Year 2000 Features

New Y2C, Y2Z, Y2P and Y2D formats, in conjunction with a new Y2PAST installation and run-time option, allow you to handle two-digit year data in the following ways:

- Set the appropriate century window for your applications (for example, 1915-2014 or 1950-2049).
- Order two-digit character, zoned decimal, packed decimal or decimal year data according to the century window using Blockset SORT or MERGE (for example, order 96 representing 1996 before 00 representing 2000 in ascending sequence, or order 00 before 96 in descending sequence).
- Transform two-digit character, zoned decimal, packed decimal or decimal year data to four-digit character year data according to the century window using OUTFIL OUTREC (for example, transform 96 to 1996 and 00 to 2000).

A new PD0 format allows you to order and transform parts of packed decimal fields (for example, month and day in date fields) using SORT, MERGE and OUTFIL.

Performance Improvements for FLR and VLR Blockset Sorts

Performance improvements for FLR and VLR Blockset sorts include the following:

- Dataspace sorting can now be used for variable-length record sort applications.
- DFSORT data processing methods have been improved.
- Dynamic storage adjustment is a new feature that allows DFSORT to automatically use more storage than the TMAXLIM value for a Blockset sort application if DFSORT determines that doing so should improve performance. New installa-

tion option DSA=n has been added to enable you to specify the dynamic storage adjustment limit.

- The upper limit for the amount of main storage that can be specified and used by DFSORT has been raised from 32M to 2000M. Specifying more main storage can provide the following benefits:
 - It allows DFSORT to sort very large data sets more efficiently.
 - It allows more sort applications to be done entirely in main storage, eliminating the need for intermediate work space and greatly reducing the EXCP counts for those applications.
 - It increases the maximum amount of data DFSORT can process in a single sort application.
- New installation option IOMAXBF=n has been added to enable you to specify the upper limit for the amount of storage to be used for SORTIN and SORTOUT data set buffers, which in turn limits the amount of data that can be transferred in a single I/O operation.
- The upper limit for the number of JCL and dynamically allocated work data sets that can be specified and used by DFSORT's Blockset technique has been raised from 32 to 100. The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application.
- Changes to the DFSORT SVC provide caching selection enhancements that improve storage control caching performance, especially for SORTIN and SORTOUT devices.
- DFSORT can now use NOEQUALS for VLR Blockset applications if EQUALS=NO is specified at installation or NOEQUALS is specified at run-time. The use of NOEQUALS can improve performance and is recommended for applications for which the order of records that collate identically need not be preserved from input to output. To minimize migration concerns, the IBM-supplied default for the ICEMAC EQUALS option is the new value VLBLKSET, which is equivalent to EQUALS=YES for VLR Blockset applications and to EQUALS=NO for all other applications.

Floating Point for SUM

FL format can now be used with the SUM control statement for short (4-byte), long (8-byte) and extended (16-byte) floating point data.

Security Improvements

Changes to the DFSORT SVC provide security improvements that bring DFSORT up to B1 security standards.

EXCPVR Processing Removed

To enhance DFSORT's protection of system integrity, EXCPVR processing will no longer be used. EXCPVR parameter values will continue to be accepted, but will have no effect on DFSORT processing. In general, the performance improvements provided by EXCPVR processing have diminished with newer technologies and will be more than offset by the performance improvements listed above. Please ignore any references to EXCPVR in this book; all such references will be deleted when the book is updated.

New Device Support for Release 13 (PTFs)

The IBM 3590 Magnetic Tape Subsystem is supported for input, output and work data sets.

Part 1. Introduction

Chapter 1. What is DFSORT?	3
DFSORT on the World Wide Web	3
DFSORT FTP Site	3
Sorting Data Sets	3
Merging Data Sets	4
Copying Data Sets	5
What Else Can You Do with DFSORT?	5
Creating and Running DFSORT Jobs	5
Writing Jobs	6
Creating Jobs Interactively	6
Running Jobs	6
Using the Sample Bookstore Data Sets	6
Creating Your Sample Input and Output Data Sets	8

Chapter 1. What is DFSORT?

DFSORT is a member of the IBM Data Facility family of products. The DFSORT licensed program is a high-performance data arranger developed by IBM for OS/390 and MVS/ESA users.

With DFSORT, you can sort, merge, and copy data sets. You can use DFSORT to do simple tasks such as alphabetizing a list of names, or you can use it to aid complex tasks such as taking inventory or running a billing system. You can also use DFSORT's record-level editing capability to perform data management tasks.

The information you manipulate with DFSORT is contained in *data sets*. The term *data set* refers to a file that contains one or more records. Any named group of records is called a *data set*. The terms *data set* and *file* are synonymous; however, for the sake of consistency, this book refers only to data sets.

A data set contains the information that you want to sort, copy, or merge. For most of the processing done by DFSORT, the whole data set is affected. However, some forms of DFSORT processing involve only certain individual records in that data set.

Data sets can be *cataloged*, which permits the data set to be referred to by name without specifying where the data set is stored. A cataloged data set should not be confused with a cataloged *procedure*. A cataloged procedure is a named collection of JCL stored in a data set, and a cataloged data set is a data set whose name is recorded by the system.

Throughout this book, the term *record* refers to a collection of related information used as a unit, such as one item in a data base or personnel data about one member of a department. The term *field* refers to a specific portion of a record used for a particular category of data. A field is the smallest addressable unit of data in a data set.

DFSORT on the World Wide Web

For articles, online books, news, tips, techniques, examples, and more, visit the DFSORT/MVS home page at URL:

<http://www.ibm.com/storage/dfsor/>

DFSORT FTP Site

You can obtain DFSORT articles and examples via anonymous FTP to:

[index.storsys.ibm.com/dfsor/mvs/](ftp://index.storsys.ibm.com/dfsor/mvs/)

Sorting Data Sets

You can use DFSORT to rearrange the records in your data sets. *Sorting* is arranging records in either ascending or descending order within a file. Table 1 on page 4 shows a sample data set of names, first sorted in ascending order, then in descending order.

What is DFSORT?

Table 1. DFSORT Arranges Information in Ascending and Descending Order

Unsorted Data Set	Sorted Ascending	Sorted Descending
Andy	Andy	Edward
Edward	Betty	Dan
Carol	Carol	Carol
Dan	Dan	Betty
Betty	Edward	Andy

The fields in the records can be in any of these formats: EBCDIC character, decimal, or binary. All of the examples in this book use the EBCDIC formatting sequence (the standard DFSORT collating sequence).

You can sort data in several different formats. Table 2 shows the most common data formats and the codes you use to specify them.

Table 2. Data Format Codes

Data Format	Code
EBCDIC (Character)	CH
Binary (Numeric)	BI
Zoned Decimal (Numeric)	ZD
Packed Decimal (Numeric)	PD

Refer to *Application Programming Guide* for complete details of the available formats.

Merging Data Sets

You can also use DFSORT to merge data sets. DFSORT *merges* data sets by combining two or more files of sorted records to form a single data set of sorted records.

Table 3. DFSORT Merges Two Data Sets into One Data Set

Data Set 1	Data Set 2	Merged Data Set
Andy	Amy	Amy
Betty	Chris	Andy
Carol	Sue	Betty
Dan		Carol
Edward		Chris
		Dan
		Edward
		Sue

The data sets you merge must be previously sorted into the same order (ascending or descending).

Copying Data Sets

DFSORT can also copy data sets without any sorting or merging taking place. You copy data sets in much the same way that you sort or merge them.

What Else Can You Do with DFSORT?

While sorting, merging, or copying data sets, you can also:

- Select a subset of records from an input data set. You can include or omit records that meet specified criteria. For example, when sorting an input data set containing records of course books from many different school departments, you can sort the books for only one department.
- Reformat records, add or delete fields, and insert blanks, constants, or binary zeros. For example, you can create an output data set that contains only certain fields from the input data set arranged differently.
- Sum the values in selected records while sorting or merging (but not while copying). In the example of a data set containing records of course books, you can use DFSORT to add up the dollar amounts of books for one school department.
- Create multiple output data sets and reports from a single pass over an input data set. For example, you can create a different output data set for the records of each department.
- Sort, merge, include or omit records according to the collating rules defined in a selected locale.
- Alter the collating sequence when sorting or merging records (but not while copying). For example, you can have the lowercase letters collate after the uppercase letters.
- Sort, merge, or copy Japanese data if the IBM Double Byte Character Set Ordering Support (DBCS Ordering) (5665-360 Licensed Program, Release 2.0 or an equivalent product) is used with DFSORT to process the records.

Creating and Running DFSORT Jobs

Processing data sets with DFSORT involves two steps:

1. Creating a DFSORT job
2. Running a DFSORT job

You can run a DFSORT job by invoking processing in a number of ways:

- With a JCL EXEC statement, using the name of the program or the name of the cataloged procedure
- With interactive panels supported under ISPF and ISMF
- Within programs written in COBOL, PL/I, or basic Assembler language

In this book, the phrase *JCL-invoked* means that the DFSORT program is initiated by a JCL EXEC statement. The phrase *dynamically invoked* means that the DFSORT program is initiated from another program. DFSORT Panels jobs are considered JCL-invoked. When you create a job interactively, DFSORT Panels supplies the JCL from the information you enter.

Writing Jobs

You can use DFSORT by writing JCL and DFSORT control statements no matter how your site has installed DFSORT. Part 1 contains instructions on writing the JCL EXEC and DFSORT program control statements.

Unless you use DFSORT Panels to prepare and submit your job, you must prepare JCL statements and DFSORT program control statements to invoke DFSORT processing. JCL statements are processed by your operating system. They describe your data sets to the operating system, and initiate DFSORT processing. DFSORT program control statements are processed by DFSORT. They describe and initiate the processing you want to do.

Creating Jobs Interactively

If your site has installed DFSORT Panels, you can perform DFSORT tasks with Interactive System Productivity Facility (ISPF) panels. For instructions on using DFSORT Panels, see *Panels Guide*.

DFSORT Panels is a productivity tool. With the panels, you can work in an online environment. You can concentrate on the sort, merge, or copy tasks, instead of on writing the correct JCL and DFSORT program control syntax. The panels automatically generate the DFSORT control statements and JCL statements from the information you enter as you go through the panels. Also, while using DFSORT Panels, you can get immediate answers to your questions online by using the HELP facility.

Running Jobs

You can run DFSORT jobs directly with a JCL EXEC statement. Or, you can call DFSORT from a COBOL, Assembler, or PL/I program.

Using the Sample Bookstore Data Sets

Before you begin, turn to Appendix A, “Using the DFSORT Sample Data Sets” on page 115. Many of the examples in this book refer to the sample bookstore data sets as the input data sets, so you should become familiar with them. The input data sets contain the data that you want arranged or sorted. You must specify an input data set for every DFSORT job you run. The sample bookstore data sets are input data sets named **`SORT.SAMPIN`** and **`SORT.SAMPADD`**.

Each record in the bookstore data sets has 12 fields (book title, author's last name, and so on). A record can be represented by one horizontal row on the page. A field can be represented by one vertical column on the page.

To sort a data set, you choose one or more fields that you want to use to order the records (arrange in ascending or descending order). These fields are called *control fields* (or, in COBOL, *keys*).

As you work through the exercises on the following pages, remember that each *entire* record is sorted, not just the control field. However, for the sake of simplicity, the figures in the text show only the control fields being discussed. The sorted records actually contain all the fields, but one page is not wide enough to show them. Appendix A, Using the DFSORT Sample Data Sets, is printed to show all the fields in each record. It is also arranged with headings and numbers that show the byte positions of each field. The numeric fields are in binary format (see Table 2

on page 4) and therefore will not appear on most terminals as they do in this book. The methods used to arrange and view the data are explained in the chapters on DFSORT functions that follow.

Table 4 shows an example of sorted fields. Notice the line of numbers above the sorted fields. These numbers represent the byte positions of those fields. You use byte positions to identify fields to DFSORT. The examples show the byte positions to help you while you are learning to use DFSORT. The byte positions do not actually appear in any of your processed data sets.

In Table 4, the first two records, which show nothing in the course department fields, are general purpose books not required for a particular course. For this example, the control field is the Course Department field.

Table 4. Sample Bookstore Data Set Sorted by Course Department in Ascending Order

Book Title		Course Department		Price	
1	75	110	114	170	173
LIVING WELL ON A SMALL BUDGET					9900
PICK'S POCKET DICTIONARY					295
INTRODUCTION TO BIOLOGY		BIOL			2350
SUPPLYING THE DEMAND		BUSIN			1925
STRATEGIC MARKETING		BUSIN			2350
COMPUTER LANGUAGES		COMP			2600
VIDEO GAME DESIGN		COMP			2199
COMPUTERS: AN INTRODUCTION		COMP			1899
NUMBERING SYSTEMS		COMP			360
SYSTEM PROGRAMMING		COMP			3195
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL			595
EDITING SOFTWARE MANUALS		ENGL			1450
MODERN ANTHOLOGY OF WOMEN POETS		ENGL			450
THE COMPLETE PROOFREADER		ENGL			625
SHORT STORIES AND TALL TALES		ENGL			1520
THE INDUSTRIAL REVOLUTION		HIST			795
EIGHTEENTH CENTURY EUROPE		HIST			1790
CRISIS OF THE MIDDLE AGES		HIST			1200
INTRODUCTION TO PSYCHOLOGY		PSYCH			2200
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH			2600

Also notice that records in Table 4 with *equally collating control fields* (in this case, the same department) appear in their original order. For example, within the Computer Science department (COMP), the title *Video Game Design* still appears before *Computers: An Introduction*.

You can control whether records with equally collating control fields appear in their original order or whether DFSORT orders them randomly. The system programmer sets defaults at installation time that you can change with some DFSORT options at run time. The examples in this book assume that the default is for records with equally collating control fields to appear in their original order.

Note: The examples used in this book are for fixed-length records only. For information on processing variable-length records, see *Application Programming Guide*.

Creating Your Sample Input and Output Data Sets

The sample bookstore data sets are the input data sets you will use for most of the examples in this book. Your system programmer created these data sets when verifying the DFSORT Panels installation. If this data set is no longer available, ask your system programmer to run the sample job, ICEDATA, to create the sample data sets, SORT.SAMPIN, SORT.SAMPOUT, SORT.SAMPADD, and SORT.BRANCH, which are used in many of the examples in this book. You must copy SORT.SAMPIN, SORT.SAMPOUT, SORT.SAMPADD, and SORT.BRANCH to your user ID before trying the examples that use them in this book.

For information on how to copy the sample data sets to your user ID, see Appendix B, “The Sample Bookstore Data Sets” on page 117.

Note: Some of the examples use data sets other than SORT.SAMPIN, SORT.SAMPOUT, SORT.SAMPADD, and SORT.BRANCH. You can either create data sets from scratch to match the ones used in the text, or else perform a similar exercise on data sets you already have.

Summary

So far in *Getting Started* you covered the following concepts:

- You can sort, copy, or merge data sets using DFSORT.
- You can either use DFSORT Panels or write the JCL EXEC and DFSORT program control statements to create and process DFSORT jobs.
- You can run DFSORT jobs directly or call DFSORT from a program.

In addition, this chapter covered how to use and read the sample bookstore data sets provided with DFSORT, and how to use the sample input and output data sets. Now continue with tutorials on how to write DFSORT control statements.

Part 2. Learning to Write JCL and DFSORT Control Statements

Chapter 2. Sorting, Merging, and Copying Data Sets	11
Sorting Data Sets	11
Sorting by Multiple Fields	13
Continuing a Statement	14
Sorting Data Sets with the JCL EXEC Statement	15
Merging Data Sets	16
Writing the MERGE Control Statement	17
Merging Data Sets with the JCL EXEC Statement	18
Copying Data Sets	19
Specifying COPY on the SORT, MERGE, or OPTION Statement	19
Copying Data Sets with the JCL EXEC Statement	20
 Chapter 3. Tailoring the Input Data Set with INCLUDE and OMIT	21
Writing the INCLUDE Statement	21
Writing the OMIT Statement	24
Allowable Comparisons for INCLUDE and OMIT	25
Writing Constants	26
Character Strings	26
Hexadecimal Strings	26
Decimal Numbers	26
 Chapter 4. Summing Records	27
Writing the SUM Statement	27
Suppressing Records with Duplicate Control Fields	29
Handling Overflow	29
 Chapter 5. Reformatting Records	31
Reformatting Records After Sorting	31
Reordering Fields to Reserve Space	32
Inserting Binary Zeros	33
Inserting Blanks	33
Inserting Constants	34
Character Strings	35
Hexadecimal Strings	35
Setting Up the Report Format	35
Reformatting Records Before Sorting	37
Using Other Statements with INREC	38
Preventing Overflow When Summing Values	40
 Chapter 6. Creating Multiple Output Data Sets and Reports	41
Creating Multiple Copies	42
Creating Multiple Output Data Sets with Unique Content	43
Creating Reports: ICETOOL vs OUTFIL	45
Using OUTFIL to Create Reports	45
Using the FNAMES and LINES Parameters	49
Using the HEADER2 Parameter	50
Using the OUTREC Parameter	52
Using the TRAILER1 Parameter	54

Chapter 7. Calling DFSORT from a Program	57
Passing Control Statements	57
Calling DFSORT from a COBOL Program	57
Sorting Records	57
Merging Records	60
Sorting with COBOL FASTSRT	63
Calling DFSORT from a PL/I Program	63
 Chapter 8. Overriding Installation Defaults	65
Specifying PARM Parameters on a JCL EXEC Statement	65
Writing an OPTION Control Statement	65
Specifying DFSPARM Parameters	66
 Chapter 9. Using DFSORT Efficiently	67
Be Generous with Main Storage	67
Use High-Speed DASD or Hiperspace	67
Eliminate Unnecessary Fields with INREC	68
Eliminate Unnecessary Records with INCLUDE or OMIT	68
Reduce File Size with STOPAFT and SKIPREC	68
Consolidate Records with SUM	68
Create Multiple Output Data Sets with OUTFIL	69
Run DFSORT with JCL	69
Use FASTSRT with COBOL	69
Avoid Options That Might Degrade Performance	69

Chapter 2. Sorting, Merging, and Copying Data Sets

This tutorial shows you how to sort, merge, and copy data sets by writing DFSORT program control statements that are processed with JCL.

DFSORT program control statements are input in the JCL used to run DFSORT. To keep the instructions simple the program control statements are covered first and the related JCL statements are explained afterward. For most of the tutorials you will concentrate on JCL-invoked DFSORT, that is, running DFSORT with JCL. Information on calling DFSORT from a program (dynamic invocation) is presented in Chapter 7, "Calling DFSORT from a Program" on page 57.

Sorting Data Sets

To run DFSORT with the JCL EXEC statement, write a SORT control statement to describe the control fields, and the order in which you want them sorted. The control statements you write are part of the input stream read from the SYSIN DD statement in the JCL.

You can use SORT with all of the other DFSORT control statements.

To write a SORT statement that sorts the bookstore records by the course department field (as shown in Table 6 on page 12):

Table 5. Steps to Create the SORT Statement to Sort by Department

Step	Action
1	Leave at least one blank, and type SORT
2	Leave at least one blank and type FIELDS=
3	Type, in parenthesis and separated by commas: <ol style="list-style-type: none">1. Where the course department field begins, relative to the beginning of the record in the bookstore data set (the first position is byte 1). The course department field begins at byte 110.2. The length of the department field in bytes. The department field is 5 bytes long.3. A code for the data format. The department field contains character data, which you specify as CH. (Table 2 on page 4 shows the codes for the most common data formats.)4. The letter A, for ascending order.

Make sure that the statement is coded between columns 2 and 71. Your control statement should look like this:

```
1  2                               71      80
   SORT  FIELDS=(110,5,CH,A)
```

Diagram illustrating the components of the SORT statement:

- 110: Beginning of department field
- 5: Length of department field
- CH: Character data
- A: Ascending order

Remember that although Figure 6 shows only certain fields, the displayed fields are not the only ones in the output data set. Your output data set will more closely resemble the fold-out of the sample bookstore data set.

Table 6. Sample Bookstore Data Set Sorted by Course Department in Ascending Order

Book Title		Course Department
1	75	110 114
LIVING WELL ON A SMALL BUDGET		
PICK'S POCKET DICTIONARY		
INTRODUCTION TO BIOLOGY		BIOL
SUPPLYING THE DEMAND		BUSIN
STRATEGIC MARKETING		BUSIN
COMPUTER LANGUAGES		COMP
VIDEO GAME DESIGN		COMP
COMPUTERS: AN INTRODUCTION		COMP
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
EDITING SOFTWARE MANUALS		ENGL
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
THE COMPLETE PROOFREADER		ENGL
SHORT STORIES AND TALL TALES		ENGL
THE INDUSTRIAL REVOLUTION		HIST
EIGHTEENTH CENTURY EUROPE		HIST
CRISES OF THE MIDDLE AGES		HIST
INTRODUCTION TO PSYCHOLOGY		PSYCH
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH

To sort the records in descending order, specify **D** instead of A. For example, to sort the prices for each book in descending order, type:

```

SORT  FIELDS=(170,4,BI,D)
      |
      |_____> Descending order
      |
      |_____> Price

```

The sort order is bytes 170 through 173 as binary data in descending sequence. Table 7 on page 13 shows the results of the sort in descending order.

Table 7. Sample Bookstore Data Set Sorted by Price in Descending Order

Book Title	Price
1	75 170 173
LIVING WELL ON A SMALL BUDGET	9900
SYSTEM PROGRAMMING	3195
COMPUTER LANGUAGES	2600
ADVANCED TOPICS IN PSYCHOANALYSIS	2600
STRATEGIC MARKETING	2350
INTRODUCTION TO BIOLOGY	2350
INTRODUCTION TO PSYCHOLOGY	2200
VIDEO GAME DESIGN	2199
SUPPLYING THE DEMAND	1925
COMPUTERS: AN INTRODUCTION	1899
EIGHTEENTH CENTURY EUROPE	1790
SHORT STORIES AND TALL TALES	1520
EDITING SOFTWARE MANUALS	1450
CRISES OF THE MIDDLE AGES	1200
THE INDUSTRIAL REVOLUTION	795
THE COMPLETE PROOFREADER	625
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	595
MODERN ANTHOLOGY OF WOMEN POETS	450
NUMBERING SYSTEMS	360
PICK'S POCKET DICTIONARY	295

Sorting by Multiple Fields

You can further sort the records in the bookstore data set by specifying multiple control fields. When you specify two or more control fields, you specify them in the order of greater to lesser priority. Note that control fields might overlap or be contained within other control fields.

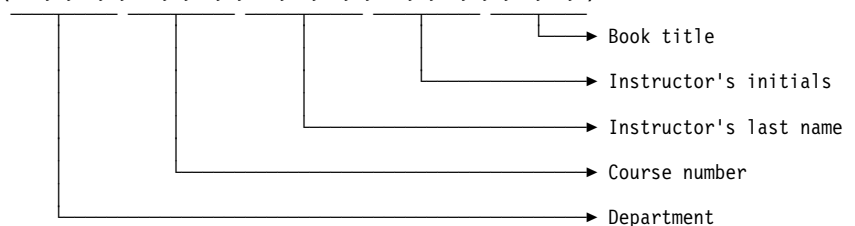
Table 8 on page 14 shows how the records would be sorted if you specified the following control fields in the order they are listed:

1. Course department
2. Course number
3. Instructor's last name
4. Instructor's initials
5. Book title.

So, if two records have the same department, they are sorted by course number. If they also have the same course number, they are sorted by instructor's last name. If they also have the same last name, they are sorted by initials. Finally, if they also have the same initials, they are sorted by title.

Specify the location, length, data format, and order for each of the control fields, as follows:

```
SORT FIELDS=(110,5,CH,A,115,5,CH,A,145,15,CH,A,160,2,CH,A,1,75,CH,A)
```



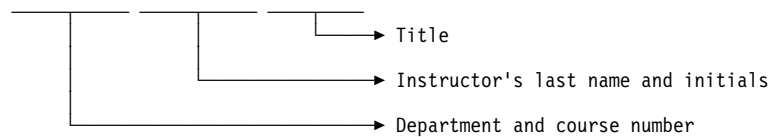
The records are sorted as shown in Table 8.

Table 8. Sample Bookstore Data Set Sorted by Multiple Fields

Book Title		Course Depart- ment	Course Number	Instructor's Last Name	Instruc- tor's Ini- tials
1	75	110 114	115 119	145 159	160 161
LIVING WELL ON A SMALL BUDGET PICK'S POCKET DICTIONARY					
INTRODUCTION TO BIOLOGY		BIOL	80521	GREENBERG	HC
STRATEGIC MARKETING		BUSIN	70124	LORCH	HH
SUPPLYING THE DEMAND		BUSIN	70251	MAXWELL	RF
NUMBERING SYSTEMS		COMP	00032	CHATTERJEE	AN
COMPUTER LANGUAGES		COMP	00032	CHATTERJEE	CL
COMPUTERS: AN INTRODUCTION		COMP	00032	CHATTERJEE	CL
SYSTEM PROGRAMMING		COMP	00103	SMITH	DC
VIDEO GAME DESIGN		COMP	00205	NEUMANN	LB
SHORT STORIES AND TALL TALES		ENGL	10054	BUCK	GR
EDITING SOFTWARE MANUALS		ENGL	10347	MADRID	MM
THE COMPLETE PROOFREADER		ENGL	10347	MADRID	MM
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL	10856	FRIEDMAN	KR
MODERN ANTHOLOGY OF WOMEN POETS		ENGL	10856	FRIEDMAN	KR
THE INDUSTRIAL REVOLUTION		HIST	50420	GOODGOLD	ST
CRISES OF THE MIDDLE AGES		HIST	50521	WILLERTON	DW
EIGHTEENTH CENTURY EUROPE		HIST	50632	BISCARDI	HR
INTRODUCTION TO PSYCHOLOGY		PSYCH	30016	ZABOSKI	RL
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH	30975	NAKATSU	FL

You can often shorten the length of control statements. You can specify fields together whenever they are next to each other and have the same data format. You can shorten this last statement by specifying the department and course number together as one field, and the instructor's last name and initials together as one field.

```
SORT FIELDS=(110,10,CH,A,145,17,CH,A,1,75,CH,A)
```



Also, if all the control fields have the same data format, you can specify the data format just once, using the **FORMAT=** parameter. For example:

```
SORT FIELDS=(110,10,A,145,17,A,1,75,A),FORMAT=CH
```

Continuing a Statement

If you cannot fit your SORT statement (or any other DFSORT control statement) between columns 2 through 71, you can continue it on the next line. If you end a line with a comma followed by a blank, DFSORT treats the next line as a continuation. The continuation can begin anywhere between columns 2 through 71.

For example:

```
SORT FIELDS=(110,10,A,145,17,A,
             1,75,A),FORMAT=CH
```

Sorting Data Sets with the JCL EXEC Statement

The job control language (JCL) you need to do a sort depends on whether you run DFSORT with the JCL EXEC statement or call DFSORT from a program. For now, concentrate on running DFSORT with the JCL EXEC statement. Information on calling DFSORT from a program is presented in Chapter 7, “Calling DFSORT from a Program” on page 57.

Your operating system uses the JCL you supply with your DFSORT program control statements to:

- Identify you as an authorized user
- Allocate the necessary resources to run your job
- Run your job
- Return information to you about the results
- Terminate your job

Unless you create your jobs with the interactive DFSORT Panels facility, you must supply JCL with every DFSORT job you submit.

Required JCL includes a JOB statement, an EXEC statement, and several DD statements. The statements you need and their exact form depend upon whether you:

- Invoke DFSORT with an EXEC statement in the input job stream, or with a system macro instruction within another program
- Choose to use EXEC statement cataloged procedures to invoke DFSORT
- Choose to specify various DFSORT options with the EXEC statement PARM parameter
- Choose to supply EXEC PARM options and DFSORT control statements together in the DFSPARM DD statement
- Choose to supply DFSORT control statements in the SYSIN DD statements
- Want to use program exits to activate routines of your own

Information on when you would choose each of the above options is detailed in *Application Programming Guide*.

The JCL statements you need for most jobs are described below.

//jobname JOB	Signals the beginning of a job. At your site, you might be required to specify information such as your name and account number on the JOB statement.
//stepname EXEC	Signals the beginning of a job step and tells the operating system what program to run. To run DFSORT, write the EXEC statement like this: //stepname EXEC PGM=SORT
//STEPLIB DD	Defines the library containing the DFSORT program. If your DFSORT program is in a system library, you can omit the STEPLIB statement.

//SYSOUT DD	Defines the output data set for messages.
//SORTIN DD	Defines the input data set.
//SORTWKdd DD	Defines a work storage data set for a sort. For most applications, one work storage data set is sufficient.
//SORTOUT DD	Defines the output data set.
//SYSIN DD	Precedes the DFSORT program control statements.

Below is some sample JCL that will run DFSORT. It assumes the input and output record lengths are the same. In Chapter 5, “Reformatting Records” on page 31, you will see how to modify the SORTOUT DD statement if the record length is changed.

```
//EXAMP JOB A492,PROGRAMMER
//SORT EXEC PGM=SORT
//STEPLIB DD DSN=A492.SM,DISP=SHR
//SYSOUT DD SYSOUT=A
//SORTIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT DD DSN=A123456.SORT.SAMPOUT,DISP=OLD
//SYSIN DD *
        SORT FIELDS=(110,10,A,145,17,A,
        1,75,A),FORMAT=CH
/*
```

Application Programming Guide contains additional information on running DFSORT with the JCL EXEC statement.

So Far

So far in this chapter you covered how to write a SORT program control statement and how to run that sort with the JCL EXEC statement. The next tutorial explains how to use the MERGE program control statement to merge two data sets.

Merging Data Sets

Generally, the reason for merging data sets is to add more records to a data set that is already sorted.

For example, assume that the bookstore data set is already sorted by course department and book title (as shown in Table 9 on page 17), and you want to update it by merging it with a data set that contains five new records, also sorted by course department and book title.

Table 9. Sample Bookstore Data Set Sorted by Course Department and Book Title

Book Title		Course Department
1	75	110 114
LIVING WELL ON A SMALL BUDGET		
PICK'S POCKET DICTIONARY		
INTRODUCTION TO BIOLOGY		BIOL
STRATEGIC MARKETING		BUSIN
SUPPLYING THE DEMAND		BUSIN
COMPUTER LANGUAGES		COMP
COMPUTERS: AN INTRODUCTION		COMP
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
VIDEO GAME DESIGN		COMP
EDITING SOFTWARE MANUALS		ENGL
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
SHORT STORIES AND TALL TALES		ENGL
THE COMPLETE PROOFREADER		ENGL
CRISES OF THE MIDDLE AGES		HIST
EIGHTEENTH CENTURY EUROPE		HIST
THE INDUSTRIAL REVOLUTION		HIST
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH
INTRODUCTION TO PSYCHOLOGY		PSYCH

For this example, use a new data set such as the one shown in Table 10.

Table 10. Five New Records Sorted by Course Department and Book Title

Book Title		Course Department
1	75	110 114
INTERNATIONAL COOKBOOK		
WORLD JOURNEYS BY TRAIN		
ARTS AND CRAFTS OF ASIA		ART
BIOCHEMISTRY		BIOL
BEHAVIORAL ANALYSIS		PSYCH

To merge data sets, you write a MERGE control statement and several JCL statements. Whenever you merge data sets, you must make sure that their records have the same format and that they have been previously sorted by the same control fields. You can merge up to 16 data sets at a time.

You can use MERGE with all of the other DFSORT control statements.

Writing the MERGE Control Statement

The format of the MERGE statement is the same as that of the SORT statement. To merge the bookstore master data set with the data set containing the five new records, write:

```

MERGE  FIELDS=(110,5,A,1,75,A),FORMAT=CH

```

Table 11 shows the merged output.

Table 11. Sample Bookstore Data Set Merged with Five New Records

Book Title	Course	Department
1	75	110 114
INTERNATIONAL COOKBOOK		
LIVING WELL ON A SMALL BUDGET		
PICK'S POCKET DICTIONARY		
WORLD JOURNEYS BY TRAIN		
ARTS AND CRAFTS OF ASIA		ART
BIOCHEMISTRY		BIOL
INTRODUCTION TO BIOLOGY		BIOL
STRATEGIC MARKETING		BUSIN
SUPPLYING THE DEMAND		BUSIN
COMPUTER LANGUAGES		COMP
COMPUTERS: AN INTRODUCTION		COMP
NUMBERING SYSTEMS		COMP
SYSTEM PROGRAMMING		COMP
VIDEO GAME DESIGN		COMP
EDITING SOFTWARE MANUALS		ENGL
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		ENGL
MODERN ANTHOLOGY OF WOMEN POETS		ENGL
SHORT STORIES AND TALL TALES		ENGL
THE COMPLETE PROOFREADER		ENGL
CRISES OF THE MIDDLE AGES		HIST
EIGHTEENTH CENTURY EUROPE		HIST
THE INDUSTRIAL REVOLUTION		HIST
ADVANCED TOPICS IN PSYCHOANALYSIS		PSYCH
BEHAVIORAL ANALYSIS		PSYCH
INTRODUCTION TO PSYCHOLOGY		PSYCH

Merging Data Sets with the JCL EXEC Statement

As in a sort, the JCL you need depends on whether you run DFSORT with the JCL EXEC statement or call it from a program. This chapter only discusses running DFSORT with the JCL EXEC statement.

The JCL needed for a merge is the same as that for a sort, with the following exceptions:

- You do *not* use the SORTWKdd DD statement.
- Instead of the SORTIN DD statement, you use SORTINnn DD statements to define the input data sets. The SORTINnn DD statements name the input data sets, and tell how many data sets will be merged. You need one SORTINnn DD statement for each data set being merged. The value nn in SORTINnn is a number from 00 to 99, indicating the number of data sets to be merged.

To merge the pre-sorted bookstore data set and the data set containing the new records, code the following JCL statements for this example. The new data set is called A123456.NEW and the sorted version of the bookstore data set is called A123456.MASTER. For this example, it is assumed that the input data sets are cataloged and that the output data set will be cataloged.

```
//EXAMP JOB A492,PROGRAMMER
//SORT EXEC PGM=SORT
//STEPLIB DD DSN=A492.SM,DISP=SHR
//SYSOUT DD SYSOUT=A
//SORTIN01 DD DSN=A123456.MASTER,DISP=OLD
//SORTIN02 DD DSN=A123456.NEW,DISP=OLD
//SORTOUT DD DSN=A123456.SORT.SAMPOUT,DISP=OLD
//SYSIN DD *
MERGE FIELDS=(110,5,A,1,75,A),FORMAT=CH
/*
```

In Chapter 7, “Calling DFSORT from a Program” on page 57, you learn how to merge data sets when calling DFSORT from a program.

So Far

So far in this chapter you covered how to write both the SORT and MERGE program control statements and how to process those control statements using the JCL EXEC statement. Now you continue with the tutorial on COPY.

Copying Data Sets

With DFSORT you can copy data sets directly without performing a sort or merge.

You can use COPY with all of the other DFSORT control statements except SUM. DFSORT can select and reformat the specific data sets you want to copy by using the control statements covered in later chapters.

You write a copy statement by specifying COPY on the SORT, MERGE, or OPTION statement.

Specifying COPY on the SORT, MERGE, or OPTION Statement

The SORT and MERGE statements change very little when you specify COPY. Just replace the information you usually put in parentheses with the word COPY:

```
SORT FIELDS=COPY
MERGE FIELDS=COPY
```

You can also specify COPY on the OPTION statement:

```
OPTION COPY
```

All three of these statements have identical results.

Copying Data Sets with the JCL EXEC Statement

The JCL for a copy application is the same as for a sort, except that you do not use the SORTWKdd DD statement.

This sample JCL will copy a data set using the OPTION COPY statement:

```
//EXAMP    JOB   A492,PROGRAMMER
//SORT     EXEC  PGM=SORT
//STEPLIB DD    DSN=A492.SM,DISP=SHR
//SYSOUT   DD    SYSOUT=A
//SORTIN   DD    DSN=A123456.SORT.SAMPIN,DISP=SHR
//SORTOUT  DD    DSN=A123456.SAMP.SORTOUT,DISP=OLD
//SYSIN    DD    *
              OPTION COPY
/*
```

You can use SORT FIELDS=COPY or MERGE FIELDS=COPY instead of OPTION COPY to produce the same results.

Summary

In this chapter of *Getting Started* you have covered the following concepts:

- Writing the SORT, COPY, or MERGE program control statements
- Using the JCL EXEC to process your sort, copy, or merge

As you continue with the tutorials, you will cover two methods of tailoring your input data set: using the INCLUDE statement and using the OMIT statement. Chapter 3, “Tailoring the Input Data Set with INCLUDE and OMIT” on page 21 covers padding and truncation rules, allowable comparison operators for INCLUDE and OMIT, and formats for writing constants and strings.

Chapter 3. Tailoring the Input Data Set with INCLUDE and OMIT

Often, you need only a subset of the records in a data set for an application. This chapter explains how to tailor the input data set by selecting specific records.

By tailoring the data set, you can increase the speed of the sort, merge, or copy. The fewer the records, the less time it takes to process them.

You tailor an input data set by:

- Using an INCLUDE control statement to collect wanted records
- Using an OMIT control statement to exclude unwanted records

Your choice of INCLUDE or OMIT depends on which is easier and more efficient to write for a given application. *You cannot use both statements together.*

You select the records you want included or omitted by comparing the contents of a field with either:

Another field For example, you can select records for which the author's last name is the same as the instructor's last name.

A constant The constant can be a character string, a decimal number, or a hexadecimal string. For example, you can select records that have the character string "HIST" in the department field.

You can also combine two conditions with logical ANDs and ORs. For example, you can select records that have either "HIST" or "PSYCH" in the department field.

INCLUDE and OMIT also offer a powerful substring search capability, and allow you to select records based on the result of bit logic tests using bit or hexadecimal masks or bit constants. Examples of these features are not shown in this book but details can be found in *Application Programming Guide*.

Writing the INCLUDE Statement

Suppose it is the end of the year and you want to sort, by title, only the books that you need to reorder for the coming year. If the number of copies sold this year for a particular book is greater than the number in stock, you can assume you need to order more copies.

To write an INCLUDE statement that selects only the books you need to order:

Table 12. Steps to Create the *INCLUDE* Statement for Books You Need to Order

Step	Action
1	Leave at least one blank and type INCLUDE
2	Leave at least one blank and type COND=
3	Type, in parentheses, and separated by commas: <ol style="list-style-type: none"> 1. The location, length, and data format of the number sold field 2. The comparison operator GT (comparison operators are shown in Figure 1 on page 22) for greater than 3. The location, length, and data format of the number-in- stock field. You can use FORMAT= when fields have the same data format.

You can select from the following comparison operators:

Comparison Operator	Meaning
EQ	Equal to
NE	Not equal to
GT	Greater than
GE	Greater than or equal to
LT	Less than
LE	Less than or equal to

Figure 1. Comparison Operators

You can place the **SORT** statement either before or after the **INCLUDE** statement. Control statements do not have to be in any specific order. However, it is good documentation practice to code them in the order in which they are processed. For a flowchart showing the order in which all the control statements are processed, see Appendix C, “Processing Order of Control Statements” on page 121.

```
INCLUDE COND=(166,4,GT,162,4,),FORMAT=BI
      |      |
      |      |-----> Number in stock
      |      |
      |      |-----> Number sold
      |
      |----->

SORT FIELDS=(1,75,CH,A)
```

This sorts the tailored data set by title in ascending order by using the **SORT** statement. Table 13 shows the sorted data set.

Table 13. Books for which Number Sold is greater than Number in Stock

Book Title		Number In Stock		Number Sold	
1	75	162	165	166	169
ADVANCED TOPICS IN PSYCHOANALYSIS		1		12	
COMPUTER LANGUAGES		5		29	
COMPUTERS: AN INTRODUCTION		20		26	
CRISES OF THE MIDDLE AGES		14		17	
EDITING SOFTWARE MANUALS		13		32	
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		2		32	
INTRODUCTION TO BIOLOGY		6		11	
MODERN ANTHOLOGY OF WOMEN POETS		1		26	
NUMBERING SYSTEMS		6		27	
STRATEGIC MARKETING		3		35	
SUPPLYING THE DEMAND		0		32	
SYSTEM PROGRAMMING		4		23	
THE COMPLETE PROOFREADER		7		19	

Suppose you want to tailor the input data set even further, to sort only the books you need to order from COR publishers. In this case, two conditions must be true:

- The number sold is greater than the number in stock.
- The book is published by COR.

To add the second condition, expand the INCLUDE statement by adding a logical AND, and compare the contents of the publisher field to the character string "COR" (see "Writing Constants" on page 26 for details how to specify constants). Because the publisher field is 4 bytes long, "COR" will be padded on the right with one blank.

```
INCLUDE COND=(166,4,BI,GT,162,4,BI,AND,106,4,CH,EQ,C'COR')
SORT FIELDS=(1,75,CH,A)
```

Table 14 shows the result.

Table 14. COR Books for which Number Sold is greater than Number in Stock

Book Title		Publisher	Number In Stock		Number Sold	
1	75	106 109	162	165	166	169
CRISES OF THE MIDDLE AGES		COR		14		17
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		COR		2		32
MODERN ANTHOLOGY OF WOMEN POETS		COR		1		26
SUPPLYING THE DEMAND		COR		0		32

As another example, you might sort only the books for courses 00032 and 10347 by writing the INCLUDE and SORT statements as follows:

```
INCLUDE COND=(115,5,CH,EQ,C'00032',OR,115,5,CH,EQ,C'10347')
SORT FIELDS=(115,5,CH,A)
```

Note: In the previous example, you cannot substitute C'32' for C'00032', because character constants are padded on the right with blanks. DFSORT uses the following rules for padding and truncation:

Padding adds fillers in data, usually zeros or blanks

Truncation deletes or omits a leading or trailing portion of a string

In comparisons, the following rules apply:

- In a field-to-field comparison, the shorter field is padded as appropriate (with blanks or zeros).
- In a field-to-constant comparison, the constant is padded or truncated to the length of the field. Decimal constants are padded or truncated on the left. Character and hexadecimal constants are padded or truncated on the right.

Writing the OMIT Statement

Suppose that you want to sort, by title, all the books used for courses but not those for general reading. In this case, you can use an OMIT statement that excludes records containing a blank in the course department field.

The format of the OMIT statement is the same as that of the INCLUDE statement. To exclude the general reading books, write:

```
OMIT COND=(110,5,CH,EQ,C' ')
SORT FIELDS=(1,75,CH,A)
```

Table 15 shows the sorted data set.

Table 15. Sorted Data Set without Books Not Required for Classes

Book Title	Course Department
1	75 110 114
ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH
COMPUTER LANGUAGES	COMP
COMPUTERS: AN INTRODUCTION	COMP
CRISES OF THE MIDDLE AGES	HIST
EDITING SOFTWARE MANUALS	ENGL
EIGHTEENTH CENTURY EUROPE	HIST
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL
INTRODUCTION TO BIOLOGY	BIOL
INTRODUCTION TO PSYCHOLOGY	PSYCH
MODERN ANTHOLOGY OF WOMEN POETS	ENGL
NUMBERING SYSTEMS	COMP
SHORT STORIES AND TALL TALES	ENGL
STRATEGIC MARKETING	BUSIN
SUPPLYING THE DEMAND	BUSIN
SYSTEM PROGRAMMING	COMP
THE COMPLETE PROOFREADER	ENGL
THE INDUSTRIAL REVOLUTION	HIST
VIDEO GAME DESIGN	COMP

Allowable Comparisons for INCLUDE and OMIT

Table 16 and Table 17 show the allowable field-to-field and field-to-constant comparisons for INCLUDE and OMIT.

Table 16. Allowable Field-to-Field Comparisons

Field Format	BI	CH	ZD	PD
BI	✓	✓		
CH	✓	✓		
ZD			✓	✓
PD			✓	✓

Table 17. Allowable Field-to-Constant Comparisons

Field Format	Character String	Hexadecimal String	Decimal Number
BI	✓	✓	
CH	✓	✓	
ZD			✓
PD			✓

For example, if you want to sort by author's name and include only those books whose author's last name begins with "M," you can compare the contents of byte 76 (the first byte of the author's last name), which is in character format, with either a character or hexadecimal string:

```
INCLUDE COND=(76,1,CH,EQ,C'M')
SORT FIELDS=(76,15,CH,A)
```

or

```
INCLUDE COND=(76,1,CH,EQ,X'D4')
SORT FIELDS=(76,15,CH,A)
```

Also, if you want to sort by number in stock only the books for which the number in stock is less than 10, you can compare the contents of the number-in-the stock field, which is in binary format, to a hexadecimal string:

```
INCLUDE COND=(162,4,BI,LT,X'0000000A')
SORT FIELDS=(162,4,BI,A)
```

Again, remember the padding and truncation rules. If you specify X'0A', the string is padded on the right instead of the left.

Writing Constants

The formats for writing character strings, hexadecimal strings, and decimal numbers are shown below.

Character Strings

The format for writing a character string is:

`C'x...x'`

where *x* is an EBCDIC character. For example, `C'FERN'`.

If you want to include a single apostrophe in the string, you must specify it as two single apostrophes. For example, *O'NEILL* must be specified as `C'O' 'NEILL'`.

Hexadecimal Strings

The format for writing a hexadecimal string is:

`X'yy...yy'`

where *yy* is a pair of hexadecimal digits. For example, `X'7FB0'`.

Decimal Numbers

The format for writing a decimal number is:

`n...n` or `±n...n`

where *n...n* is a decimal digit. Examples are 24, +24, and -24.

Decimal numbers must not contain commas or decimal points.

Summary

This chapter covered two ways to tailor the input data set to make processing more efficient. You wrote `INCLUDE` and `OMIT` statements and read about allowable comparison operators and the formats for writing constants in the program control statement.

Chapter 4. Summing Records

Suppose that the English department wants to know the total price of books for all its courses. You can tailor the file to include only records for the English department by using the INCLUDE statement, and add the book prices together by using the SORT and SUM statements.

On the SUM control statement, you specify one or more numeric fields that are to be summed whenever records have equally collating control fields (control fields are specified on the SORT statement). The numeric fields can be in binary, packed decimal, or zoned decimal format.

To sum the prices for all the records for the English department, specify the price field on the SUM statement and the department field on the SORT statement. By the time SUM and SORT are processed, INCLUDE has already tailored the file to contain only the records for the English department, making the department field equal for all the records, and allowing the prices to be summed. (For a flowchart showing the order in which the INCLUDE, SUM, and SORT statements are processed, see Appendix C, "Processing Order of Control Statements" on page 121.)

When you sum records, keep in mind that two types of fields are involved:

Control fields specified on the SORT statement

Summary fields specified on the SUM statement

The contents of the summary fields are summed only when the contents of the control fields are of the same data type. See Table 2 on page 4.

Writing the SUM Statement

To write a SUM statement that sums the prices for the English department:

Table 18. Steps to Create the SUM Statement for Prices

Step	Action
1	Leave at least one blank and type SUM
2	Leave at least one blank and type FIELDS=
3	Type, in parentheses and separated by commas, the location, length, and data format of the price field.

The INCLUDE, SORT, and SUM statements are shown below:

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL')
SORT FIELDS=(110,5,CH,A)
SUM FIELDS=(170,4,BI)
```

└──────────┐
└──────────┘ → Price

When the prices are summed, the final sum appears in the price field of one record, and the other records are deleted. Therefore, the result (shown in Table 19 on page 28) is only one record, containing the sum. You can control which record appears if you specify that records keep their original order. For the examples, the default is for records with equally collating control fields to appear in their original

Summing Records

order. When summing records keeping the original order, DFSORT chooses the first record to contain the sum.

Table 19. Sum of Prices for English Department

Book Title	Course Department	Price
1 75	110 114	170 173
INKLINGS: AN ANTHOLOGY OF YOUNG POETS ¹	ENGL ²	4640 ³

Note:

- ¹ Some of the fields in your summation record might not be meaningful, such as the book title field in Table 19. You could use the OMIT statement to omit this field. In the next chapter, you will learn two ways to leave out fields that are not meaningful.
- ² Specified as a control field.
- ³ Specified as a summary field.

Suppose now that the English department wants to know the total price of books for *each* of its courses. In this case, you still select only the English department's records using INCLUDE, and specify the price field on the SUM statement, but you specify the *course number* on the SORT statement.

```
INCLUDE COND=(110,5,CH,EQ,C'ENGL')
SORT FIELDS=(115,5,CH,A)
SUM FIELDS=(170,4,BI)
```

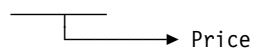


Table 20 shows the result, one record per course.

Table 20. Sum of Prices for English Department

Book Title	Course Number	Price
1 75	115 119 170 173	
SHORT STORIES AND TALL TALES	10054	1520
EDITING SOFTWARE MANUALS	10347	2075
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	10856	1045

For an example using two summary fields, assume that for inventory purposes you want to sum separately the number of books in stock and the number sold for each of the publishers.

For this application, specify the publisher as the control field on the SORT statement and the number in stock and number sold as summary fields on the SUM statement.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,166,4),FORMAT=BI
```

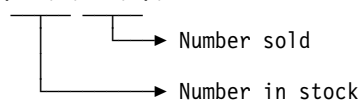


Table 21 on page 29 shows the result, one record per publisher.

Table 21. Sum of Number in Stock and Number Sold for Each Publisher

Book Title		Publisher		Number In Stock		Number Sold	
1	75	106	109	162	165	166	169
LIVING WELL ON A SMALL BUDGET		COR		103		161	
COMPUTER LANGUAGES		FERN		19		87	
VIDEO GAME DESIGN		VALD		42		97	
COMPUTERS: AN INTRODUCTION		WETH		62		79	

Suppressing Records with Duplicate Control Fields

Apart from summing values, you can also use SUM to delete records with duplicate control fields.

For example, you might want to list the publishers in ascending order, with each publisher appearing only once. If you use only the SORT statement, COR appears seven times (because seven books in the file are published by COR), FERN appears four times, VALD five times, and WETH four times.

By specifying FIELDS=NONE on the SUM statement, as shown below, DFSORT writes only one record per publisher:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=NONE
```

Table 22 shows the result.

Table 22. List of Publishers, Deleting Duplicates

Book Title		Publisher
1	75	106 109
LIVING WELL ON A SMALL BUDGET		COR
COMPUTER LANGUAGES		FERN
VIDEO GAME DESIGN		VALD
COMPUTERS: AN INTRODUCTION		WETH

Handling Overflow

When a sum becomes larger than the space available for it, *overflow* occurs. For example, if a 2-byte binary field (unsigned) contains X'FFFF' and you add X'0001' to it, overflow occurs, because the sum requires more than two bytes.

```
FFFF
0001
10000
```

If overflow occurs, the two records involved are not added together. That is, the contents of the records are left untouched, neither record is deleted, and the records are still available to be summed. Overflow does not prevent further summary.

Summing Records

In some cases, you can correct overflow by padding the summary fields with zeros, using the INREC control statement. “Preventing Overflow When Summing Values” on page 40 shows you how to do this.

Summary

This chapter covered summing records in your data set. It explained how to use the SUM statement to sum records with equal control fields, and how to suppress any records with duplicate control fields. Now, you continue with tutorials about using OUTREC and INREC to reformat your data sets.

Chapter 5. Reformatting Records

You can reformat records in your data sets by using the OUTREC and INREC control statements. With OUTREC and INREC, you can:

- Delete fields.
- Reorder fields.
- Insert separators (blanks, zeros, or constants).

The difference between the two DFSORT control statements is that OUTREC reformats records *after* they are sorted, copied, or merged, whereas INREC reformats records *before* they are sorted, copied, or merged. This has an effect on other control statements. See “Using Other Statements with INREC” on page 38 for information about how INREC affects other control statements.

INREC and OUTREC perform the same functions. When deciding which to use, remember their processing order. In general:

- If you are deleting fields, try to use INREC because shorter records take less time to sort, merge, or copy (INREC reformats the records before they are processed).
- If you are going to insert separators, use OUTREC because OUTREC inserts the separators into the records after they are processed.
- If you are reordering fields, you can use either control statement because reordering fields does not affect the record length.

Reformatting Records After Sorting

In the last chapter, you used the SUM statement to sum the price of the books in stock and the books sold for each publisher. Now, using the OUTREC statement, you can delete all the fields that are not needed for the application; in other words, fields whose contents are not meaningful in a summation record. Only the publisher, number-in-stock, and number-sold fields are written, reducing the output record length to 12 bytes.

To write the OUTREC statement:

Table 23. Steps to Create the OUTREC Statement for Reformatting Records

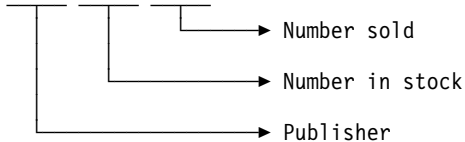
Step	Action
1	Leave at least one blank, and type OUTREC
2	Leave at least one blank, and type FIELDS=
3	Type, in parentheses, and separated by commas: <ol style="list-style-type: none"> 1. The location and length of the publisher field 2. The location and length of the number-in-stock field 3. The location and length of the number-sold field.

Because the number-in-stock and number-sold fields are next to each other, you can also specify them together as one field. They do not need to have the same data format.

Reformatting Records

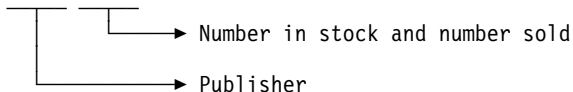
Note that on the OUTREC statement you do *not* specify the data format.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,4,166,4)
```



Or:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,162,8)
```



Note: If you use INREC or OUTREC to change the record length, be sure to specify the *final* record length on the SORTOUT DD statement using the DCB parameter (the DCB parameter is shown coded later in this chapter). The final length is either:

- The INREC length if you are using just INREC
- The OUTREC length if you are using just OUTREC or both INREC and OUTREC

For example, this change is required for the above statements:

```
//SORTOUT DD DSN=A123456.OUTF12,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1)),UNIT=SYSDA,
//          DCB=LRECL=12
```

Table 24 shows the output.

Table 24. Writing Only Publisher, Number In Stock, and Number Sold Fields

Publisher	Number In Stock	Number Sold
1 4	5 8	9 12
COR	103	161
FERN	19	87
VALD	42	97
WETH	62	79

Reordering Fields to Reserve Space

The fields always appear in the order in which you specify them. Therefore, if you want the number sold to appear before the number in stock, as shown in Table 25 on page 33, you reverse their order on the OUTREC statement.

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4)
```

Table 25. Reordering the Fields

Publisher	Number Sold	Number In Stock
1 4	5 8	9 12
COR	161	103
FERN	87	19
VALD	97	42
WETH	79	62

Inserting Binary Zeros

Building on the last example, assume you want to reformat the records to include a new 4-byte binary field after the number in stock (beginning at byte 13). In this case, you can insert binary zeros as place holders for the new field (to be filled in with data at a later date). You can use **Z** or **1Z** to specify a single binary zero.

To insert the zeros, write 4Z after the last field:

```
SORT FIELDS=(106,4,CH,A)
SUM FIELDS=(162,4,BI,166,4,BI)
OUTREC FIELDS=(106,4,166,4,162,4,4Z)
```

This time, you must specify on the SORTOUT DD statement that the new record length is 16 bytes:

```
//SORTOUT DD DSN=A123456.OUTF16,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1)),UNIT=SYSDA,
//          DCB=LRECL=16
```

Table 26 shows the result.

Table 26. Inserting Binary Zeros

Publisher	Number Sold	Number In Stock	X'0...0'
1 4	5 8	9 12	13 16
COR	161	103	0...0
FERN	87	19	0...0
VALD	97	42	0...0
WETH	79	62	0...0

Inserting Blanks

If an output data set contains *only character data*, you can print it by writing the SORTOUT DD statement as follows:

```
//SORTOUT DD SYSOUT=A
```

You can make the printout more legible by using the OUTREC statement to separate the fields with blanks and to create margins. You can insert blanks before, between, or after fields. You can use **X** or **1X** to specify a single blank.

Reformatting Records

For example, assume you want to print just the publisher and title fields, with the publisher field appearing first. Because most of the publishers' names fill up the entire 4-byte publisher field, the publishers' names will run into the titles if you do not separate the two fields with blanks. Also, without a margin, the publishers' names will begin at the edge of the paper.

The printout can be made more legible by separating the fields with 10 blanks and creating a margin of 20 blanks.

To insert the blanks, specify 10X between the two fields, and 20X before the first field. The SORT statement sorts the records by title in ascending order (remember that SORT or MERGE is always required).

```
SORT  FIELDS=(1,75,CH,A)
OUTREC FIELDS=(20X,106,4,10X,1,75)
```

Table 27 shows the result.

Table 27. Output After Inserting Blanks

Publisher						Book Title	
1	20	21	24	25	34	35	190
(20 Blanks)		(10 Blanks)					
		FERN				ADVANCED TOPICS IN PSYCHOANALYSIS	
		FERN				COMPUTER LANGUAGES	
		WETH				COMPUTERS: AN INTRODUCTION	
		COR				CRISES OF THE MIDDLE AGES	
		VALD				EDITING SOFTWARE MANUALS	
		WETH				EIGHTEENTH CENTURY EUROPE	
		COR				INKLINGS: AN ANTHOLOGY OF YOUNG POETS	
		VALD				INTRODUCTION TO BIOLOGY	
		COR				INTRODUCTION TO PSYCHOLOGY	
		COR				LIVING WELL ON A SMALL BUDGET	
		COR				MODERN ANTHOLOGY OF WOMEN POETS	
		FERN				NUMBERING SYSTEMS	
		COR				PICK'S POCKET DICTIONARY	
		VALD				SHORT STORIES AND TALL TALES	
		VALD				STRATEGIC MARKETING	
		COR				SUPPLYING THE DEMAND	
		WETH				SYSTEM PROGRAMMING	
		FERN				THE COMPLETE PROOFREADER	
		WETH				THE INDUSTRIAL REVOLUTION	
		VALD				VIDEO GAME DESIGN	

Inserting Constants

In addition to making the printout more legible, OUTREC can also be used to set up a very basic report format by inserting constants. ICETOOL's DISPLAY operator or the OUTFIL control statement can be used to create complex reports as you will see in later examples. The formats for writing constants are shown below.

Character Strings

The format for writing a character string is:

`C'x...x'`

where *x* is an EBCDIC character. For example, `C'FERN'`.

The format for writing character string repetition is:

`nC'x...x'`

where *n* can be from 1 to 4095; *n* repetitions of the character string constant (`C'x...x'`) are inserted into the reformatted input records. If *n* is omitted, 1 is used instead.

If you want to include a single apostrophe in the string, you must specify it as two single apostrophes. For example, *O'NEILL* must be specified as `C'O' 'NEILL'`.

Hexadecimal Strings

The format for writing a hexadecimal string is:

`X'yy...yy'`

where *yy* is a pair of hexadecimal digits. For example, `X'7FB0'`.

The format for writing hexadecimal string repetition is:

`X'yy...yy'`

where *n* can be from 1 to 4095. *n* repetitions of the hexadecimal string constant (`X'yy...yy'`) are inserted in the reformatted input records. If *n* is omitted, 1 is used.

Setting Up the Report Format

To produce a very basic report of the publisher's names and author's names from the bookstore data set, you can use OUTREC to put in "Publisher is" and "Author is" as character separators.

To write the OUTREC statement:

Table 28. Steps to Write the OUTREC Statement

Step	Action
1	Leave at least one blank, and type OUTREC
2	Leave at least one blank, and type FIELDS=
3	Type, in parenthesis: <ol style="list-style-type: none"> 1. The column you want the first term to start in, which is 11, followed by a colon. The colon indicates column alignment, and must be followed by a field or separator. OUTREC automatically inserts blanks before this column, or from the end of the previous field up to this column. 2. The letter C. 3. The term Publisher is in single quotes and followed by a comma. Make sure that there is one space after the is and before the single quote. Otherwise, the first name will look like a continuation of the word is. (Alternatively, you can use X for the space.) 4. The location (106) and length (4) of the publisher field, each followed by a comma. 5. The column you want the second term to start in, which is 31, followed by a colon. 6. The term Author is in single quotes (with an extra space), followed by a comma. 7. The location (91) and length (15) of the author's-first-name field, each followed by a comma. 8. The letter X, for one blank, followed by a comma. 9. The location of the author's-last-name field (76), followed by a comma, and the length of the field (15).

The statement is shown below:

```
OPTION COPY
OUTREC FIELDS=(11:C'Publisher is ',106,4,
31:C'Author is ',91,15,X,76,15)
```

The result is shown in Table 29 on page 37.

Table 29. Output of a Report

								Author's First Name		Author's Last Name	
1	10	11	22	24	27	31	39	41	55	57	71
(10 blanks)		Publisher is	FERN			Author is		ROBERT		MURRAY	
		Publisher is	COR			Author is		FRANK		DEWAN	
		Publisher is	COR			Author is		TOM		MILLER	
		Publisher is	VALD			Author is		LORI		RASMUSSEN	
		Publisher is	COR			Author is		KAREN		WILDE	
		Publisher is	WETH			Author is		JOKHI		DINSHAW	
		Publisher is	COR			Author is		CAROL		GUSTLIN	
		Publisher is	VALD			Author is		VICTOR		OJALVO	
		Publisher is	FERN			Author is		WILLIAM		BAYLESS	
		Publisher is	VALD			Author is		MARK		YAEGER	
		Publisher is	WETH			Author is		DON		GROSS	
		Publisher is	COR			Author is		PETER		COWARD	
		Publisher is	COR			Author is		LINDA		DUZET	
		Publisher is	FERN			Author is		ANN		GREEN	
		Publisher is	WETH			Author is		RAUL		CAUDILLO	
		Publisher is	VALD			Author is		LILIANA		AVRIL	
		Publisher is	VALD			Author is		CHIEN		WU	
		Publisher is	FERN			Author is		DIANNE		OSTOICH	
		Publisher is	WETH			Author is		ALICE		MUNGER	
		Publisher is	COR			Author is		GREG		BENDER	

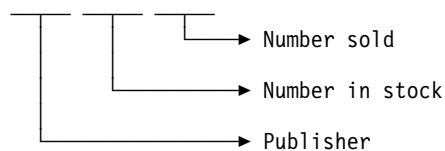
So Far

So far this chapter has covered how the OUTREC statement can define only certain fields to go in the records of your output data set. This chapter has also covered inserting binary zeros as place holders, and using blanks and constants to make a printout of the output data set more readable.

Reformatting Records Before Sorting

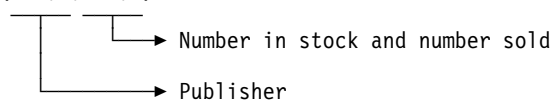
The INREC statement has the same format as the OUTREC statement. Therefore, in the first example of “Reformatting Records After Sorting” on page 31, where you used OUTREC to write only the publisher, number-in-stock, and number-sold fields, you could use INREC instead, as shown below.

```
INREC FIELDS=(106,4,162,4,166,4)
```



Or:

```
INREC FIELDS=(106,4,162,8)
```



Using Other Statements with INREC

Because INREC reformats the records *before* they are sorted, the SORT and SUM statements must refer to the *reformatted* records as they will appear in the output data set.

Thus, after INREC, the input records for the control statement in the previous section are 12 bytes long (see Table 24 on page 32 for an example).

You write the SORT and SUM statements to process the byte positions in the output data set:

```
INREC FIELDS=(106,4,162,8)
SORT FIELDS=(1,4,CH,A)
SUM FIELDS=(5,4,BI,9,4,BI)
```

Table 30 shows the result.

Table 30. Using INREC to Write Only Publisher, Number in Stock, and Number Sold

Publisher	Number In Stock	Number Sold
1 4	5 8	9 12
COR	103	161
FERN	19	87
VALD	42	97
WETH	62	79

As the flowchart in Appendix C, “Processing Order of Control Statements” on page 121 shows, DFSORT processes the INREC statement *before* SORT, SUM, and OUTREC, but *after* INCLUDE and OMIT. Therefore, when used with the INREC statement, SORT, SUM, and OUTREC must refer to the *reformatted* records, and INCLUDE and OMIT must refer to the *original* records.

DFSORT processes the OUTFIL statements after the INREC and OUTREC statements. Therefore, OUTFIL must refer to the reformatted records produced by OUTREC if specified, or to the reformatted records produced by INREC if it is specified without OUTREC.

Suppose you want to select and reformat specific records from the file shown in Table 31:

Table 31. Bookstore Data Set as a Source for a Copy Application

Book Title	Course Department	Price
1	75	110 114 170 173
LIVING WELL ON A SMALL BUDGET		9900
PICK'S POCKET DICTIONARY		295
INTRODUCTION TO BIOLOGY	BIOL	2350
SUPPLYING THE DEMAND	BUSIN	1925
STRATEGIC MARKETING	BUSIN	2350
COMPUTER LANGUAGES	COMP	2600
COMPUTERS: AN INTRODUCTION	COMP	1899
NUMBERING SYSTEMS	COMP	360
SYSTEM PROGRAMMING	COMP	3195
VIDEO GAME DESIGN	COMP	2199
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	ENGL	595
EDITING SOFTWARE MANUALS	ENGL	1450
MODERN ANTHOLOGY OF WOMEN POETS	ENGL	450
THE COMPLETE PROOFREADER	ENGL	625
SHORT STORIES AND TALL TALES	ENGL	1520
THE INDUSTRIAL REVOLUTION	HIST	795
EIGHTEENTH CENTURY EUROPE	HIST	1790
CRISES OF THE MIDDLE AGES	HIST	1200
INTRODUCTION TO PSYCHOLOGY	PSYCH	2200
ADVANCED TOPICS IN PSYCHOANALYSIS	PSYCH	2600

From the complete data set, you want a copy of just the reading list (without the prices) for the computer department.

Use the INCLUDE statement to select only departments equal to "COMP," add the INREC statement to include only the title and department fields, and use the OPTION statement to specify the copy function. The statements look like this:

```
INCLUDE COND=(110,5,CH,EQ,C'COMP')
INREC FIELDS=(1,114)
OPTION COPY
```

Table 32 shows the copy of the data set.

Table 32. List of Computer Texts Copied from Bookstore Data Set

Book Title	Course Department
1	75
COMPUTER LANGUAGES	COMP
COMPUTERS: AN INTRODUCTION	COMP
NUMBERING SYSTEMS	COMP
SYSTEM PROGRAMMING	COMP
VIDEO GAME DESIGN	COMP

Preventing Overflow When Summing Values

In some cases, you can prevent overflow by using INREC to pad summary fields with zeros. However, this method cannot be used for negative fixed-point binary data, because padding with zeros rather than with ones would change the sign.

If the summary fields in Table 30 on page 38 were overflowing, you could pad each of them on the *left* with 4 bytes (binary fields must be 2, 4, or 8 bytes long), as shown in Table 33.

Table 33. Padding Summary Fields

Publisher	X'0...0'	Number In Stock	X'0...0'	Number Sold
1 4	5 8	9 12	13 16	17 20
COR		103		161
FERN		19		87
VALD		42		97
WETH		62		79

```
INREC FIELDS=(106,4,4Z,162,4,4Z,166,4)
```

```
SORT FIELDS=(1,4,CH,A)
```

```
SUM FIELDS=(5,8,BI,13,8,BI)
```

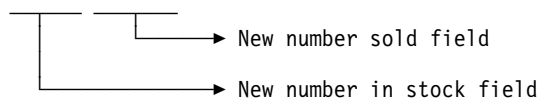


Table 34 shows the output records, each 20 bytes long.

Table 34. Padding Summary Fields

Publisher	Number In Stock	Number Sold
1 4	5 12	13 20
COR	103	161
FERN	19	87
VALD	42	97
WETH	62	79

Note: You cannot use the OUTREC statement to prevent overflow, because it is processed *after* summarization.

Summary

This chapter covered using INREC and OUTREC to reformat data sets. You can delete fields, insert blanks, zeros or constants, and reorder fields with both of these control statements. These two control statements help improve the appearance of your output.

Chapter 6. Creating Multiple Output Data Sets and Reports

You can create multiple output data sets and reports from a single pass over sorted, merged, or copied input using UTFIL control statements. With UTFIL you can:

- Create multiple output data sets from a single pass over the input data set. These data sets can have the same or different records and field arrangements because each UTFIL statement can specify one or more UTFIL ddnames and can have its own INCLUDE or OMIT and OUTREC parameters as well as report parameters.
- Create detailed reports with three levels (report, page, and section) of report elements such as:
 - Headers
 - Trailers
 - Totals
 - Counts
 - Maximums
 - Minimums
 - Averages
 - Page control via line counters
- Edit numeric fields to printable hexadecimal characters, or printable decimal characters with thousands separator, decimal point, leading or suppressed zeros, signs, and so on. You can select one or more of UTFIL's twenty-six pre-defined editing masks, or specify your own user-defined editing masks, which allows virtually unlimited selection of numeric formats.
- Change fields that match selected character, hexadecimal, or bit constants to selected character or hexadecimal constants. This provides a powerful function for substituting meaningful names, words and phrases for otherwise cryptic values.
- Create summary reports that show report elements such as headers, trailers and statistics, but not the actual data records.
- Select sequential groups of records for output data sets (STARTREC and ENDREC parameters), split records evenly between output data sets (SPLIT parameter), and create fixed-length records from variable-length records (CONVERT parameter). Examples with these parameters are not shown here, but details can be found in *Application Programming Guide*.

All of the data sets specified for a particular UTFIL statement are processed in a similar way and thus are referred to as an **UTFIL group**.

To better illustrate UTFIL's capabilities, the last example in this chapter uses an extra data set, SORT.SAMPADD, concatenated with SORT.SAMPIN, the data set that was used in previous examples. Appendix B, "The Sample Bookstore Data Sets" on page 117 and Appendix A, "Using the DFSORT Sample Data Sets" on page 115 show you how to use the sample data sets and the contents of each data set.

Creating Multiple Copies

Suppose you want to create backups for your data set; on DASD for the local site and on tapes for the remote sites. You can do this by using OUTFIL and the FNames parameter with an OPTION COPY statement. With the FNames parameter, you define the output data sets with unique DD statements in the JCL before writing your DFSORT control statements. Sample JCL and the OPTION COPY statement are shown below.

```
//COPY JOB A492,PROGRAMMER
//SORT EXEC PGM=SORT
//STEPLIB DD DSN=A492.SM,DISP=SHR
//SYSOUT DD SYSOUT=A
//SORTIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//BACKUP DD DSN=A123456.BOOKS.BACKUP,DISP=OLD
//NEWYORK DD DSN=BOOKS,UNIT=3490,DISP=(,KEEP),VOL=SER=REMOT1,LABEL=(,SL)
//SANJOSE DD DSN=BOOKS,UNIT=3490,DISP=(,KEEP),VOL=SER=REMOT2,LABEL=(,SL)
//SYSIN DD *
        OPTION COPY
```

Note: The JCL above includes two tape data sets. Substitute your own tape data set information if you want to run this example.

To write the OUTFIL statement:

Table 35. Creating the OUTFIL Statement for the Multiple Output Data Set Job	
Step	Action
1	Leave at least one blank and type OUTFIL
2	Leave at least one blank and type FNames=
3	Type, in parentheses and separated by commas, each output data set DD name. In this example, they are BACKUP , NEWYORK , and SANJOSE .

The OUTFIL statement is shown below:

```
        OUTFIL FNames=(BACKUP,NEWYORK,SANJOSE)
/*
```

This statement specifies three backup copies of your data set: one on-site DASD and two tapes that can be sent to remote sites.

You can also complete this task using the FILES parameter of OUTFIL. With FILES, you assign SORTOFd or SORTOFdd DD statements instead of naming unique DD statements. Sample JCL and DFSORT control statements using FILES are shown below.

```
//COPY JOB A492,PROGRAMMER
//SORT EXEC PGM=SORT
//STEPLIB DD DSN=A492.SM,DISP=SHR
//SYSOUT DD SYSOUT=A
//SORTIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//SORTOF1 DD DSN=A123456.BOOKS.BACKUP,DISP=OLD
//SORTOF2 DD DSN=BOOKS,UNIT=3490,DISP=(,KEEP),VOL=SER=REMOT1
//SORTOF3 DD DSN=BOOKS,UNIT=3490,DISP=(,KEEP),VOL=SER=REMOT2
//SYSIN DD *
    OPTION COPY
    OUTFIL FILES=(1,2,3)
/*
```

Both FNames and FILES create the same output, but FNames lets you keep track of your data more easily by defining specific DD names instead of one or two suffix characters. FNames is used in all of the multiple output examples in the rest of this chapter.

Creating Multiple Output Data Sets with Unique Content

You can use INCLUDE or OMIT as OUTFIL parameters to create multiple output data sets with unique content. While INCLUDE and OMIT statements apply to all input records, the INCLUDE and OMIT parameters only apply to records received by OUTFIL from previous DFSORT processing (such as SORT and INCLUDE control statements). These records are the OUTFIL input records. In addition, different INCLUDE or OMIT parameters can be used with each OUTFIL statement.

Any logical expression that is valid for the COND parameter of the INCLUDE or OMIT control statement is valid for the corresponding parameter. However, you can not specify FORMAT= with the INCLUDE and OMIT parameters as you can with the control statements.

In this example, the bookstore data set is sorted by title and the books for each department are included in separate output data sets. The books that don't belong to any department are placed in a separate data set using the SAVE parameter. The SAVE parameter operates in a global fashion over all of the other OUTFIL statements for which SAVE is not specified and allows you to keep any OUTFIL input records that would not be kept otherwise. The new OUTFIL parameters used in this example are:

INCLUDE selects the records to be included in the data sets for this OUTFIL group (OMIT selects the records to be omitted from the data sets for this OUTFIL group)

SAVE specifies that OUTFIL input records not included for any other OUTFIL group are to be included in the data sets for this OUTFIL group

To write the OUTFIL statements for this job:

Table 36. Creating the OUTFIL Statement for the Unique Output Data Set Job

Step	Action
1	Leave at least one blank and type OUTFIL
2	Leave at least one blank and type FNAMES=
3	Type the name of your output data set definition and a comma
4	Type INCLUDE=
5	Type, in parentheses and separated by commas: <ol style="list-style-type: none"> 1. The location of the department field 2. The length of the department field 3. The data format of the department field 4. The comparison operator EQ 5. The character constant for the department; for example, C'ENGL'
6	Repeat steps 1-5 for each department
7	For the books without an entry in the department field, repeat only steps 1-3. Then type SAVE .

The JCL and DFSORT control statements for this example are shown below.

```
//MULTI JOB A492,PROGRAMMER
//SORT EXEC PGM=SORT
//STEPLIB DD DSN=A492.SM,DISP=SHR
//SYSOUT DD SYSOUT=A
//SORTIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//ENGL DD DSN=A123456.SORT.ENGL,DISP=OLD
//BIOL DD DSN=A123456.SORT.BIOL,DISP=OLD
//BUSIN DD DSN=A123456.SORT.BUSIN,DISP=OLD
//COMP DD DSN=A123456.SORT.COMP,DISP=OLD
//HIST DD DSN=A123456.SORT.HIST,DISP=OLD
//PSYCH DD DSN=A123456.SORT.PSYCH,DISP=OLD
//EXTRA DD DSN=A123456.SORT.EXTRA,DISP=OLD
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN DD *
  SORT FIELDS=(1,75,CH,A)
  OUTFIL FNAMES=ENGL,INCLUDE=(110,5,CH,EQ,C'ENGL')
  OUTFIL FNAMES=BIOL,INCLUDE=(110,5,CH,EQ,C'BIOL')
  OUTFIL FNAMES=BUSIN,INCLUDE=(110,5,CH,EQ,C'BUSIN')
  OUTFIL FNAMES=COMP,INCLUDE=(110,5,CH,EQ,C'COMP')
  OUTFIL FNAMES=HIST,INCLUDE=(110,5,CH,EQ,C'HIST')
  OUTFIL FNAMES=PSYCH,INCLUDE=(110,5,CH,EQ,C'PSYCH')
  OUTFIL FNAMES=EXTRA,SAVE
/*
```

This example only shows one way to use INCLUDE as a parameter with OUTFIL. For a more detailed look at the many ways INCLUDE and OMIT can be used, see Chapter 3, “Tailoring the Input Data Set with INCLUDE and OMIT” on page 21.

So Far

So far, you have learned how to copy to multiple output data sets and to create multiple output data sets with unique content using OUTFIL, FNames, FILES, INCLUDE, and SAVE. Next, you will learn about creating reports with DFSORT.

Creating Reports: ICETOOL vs OUTFIL

Both the OUTFIL statement and ICETOOL's DISPLAY operator can be used to create reports. While each performs some reporting functions that the other does not, in general the difference between them is one of control and effort. With OUTFIL, you have more control over the appearance of reports, but considerable effort may be required on your part to specify in detail where every piece of information is to appear in the report and how it is to look. With ICETOOL, much of the work of determining where information is to appear in the report and how it is to look is done for you, but you have less control over the appearance of the reports.

ICETOOL should be your primary choice for creating reports since it is easier to use than OUTFIL. But remember that OUTFIL is available if you need any of its specific reporting features, or more control of the appearance of reports than ICETOOL gives you. See Chapter 10, "Using the ICETOOL Utility" on page 73 for examples of using DISPLAY to create reports.

Using OUTFIL to Create Reports

You can use OUTFIL to format your output into an easy-to-read report by using:

- The OUTREC parameter to arrange your OUTFIL input records as data records for the report
- Report parameters (HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, NODETAIL and LINES) to include report elements such as titles, column headings, totals, counts, maximums, minimums, and averages, in report records for the report. This example does not use HEADER1, TRAILER2, SECTIONS or NODETAIL, but details of these parameters (as well as all other OUTFIL parameters and subparameters) can be found in *Application Programming Guide*.

You can include any or all of these in your report:

- A cover sheet (report header)
- A header at the top of each page (page header)
- A trailer at the bottom of each page (page trailer)
- A header at the start of each section (section header)
- A trailer at the start of each section (section trailer)
- A summary sheet (report trailer)

While the OUTREC statement applies to all input records, the OUTREC parameter only applies to the OUTFIL input records. Different OUTREC parameters can be used with each OUTFIL statement, as can different headers, trailers, and so on.

All of the fields and separators that are valid for the FIELDS parameter of the OUTREC statement are also valid for the OUTREC parameter. However, the

OUTREC parameter offers additional features such as edit masks, edit patterns, and table lookup and change, that are not available with the OUTREC statement.

The new OUTFIL parameters used in this example are:

LINES	specifies the number of lines per page to be used for the report (the default is 60 lines per page)
HEADER2	specifies a header that appears at the top of each page of the report, except for the cover sheet and summary sheet
OUTREC	specifies how the records in the data sets for this OUTFIL group are to be reformatted as data records for the report
TRAILER2	specifies a trailer that appears by itself as the last page of the report

When you create an OUTFIL report, the length for the data records must be equal to or greater than the maximum report record length. You can use the OUTREC parameter to force a length for the data records that is longer than any report record by adding a blank to the data records at a point beyond the report record length. For example, if your data records are 40 bytes, but your longest report record is 60 bytes, you could use an OUTREC parameter such as:

```
OUTREC=(1,40,80:X)
```

You can also use the OUTREC parameter to convert numeric fields to printable hexadecimal characters, or printable decimal characters with sign control (for example, + for positive numbers and – or () for negative numbers) and editing control (for example, commas every n digits, decimal point, leading \$, suppressed or non-suppressed leading zeros, and so on). You can edit BI, FI, PD, ZD, and FS/CSF format fields using either the pre-defined M0-M25 edit mask subparameters or specific edit patterns you define with the EDIT subparameter. You can use the M0 edit mask automatically by specifying a format without a specific edit mask or edit pattern.

The 26 pre-defined edit masks can be represented as follows:

Table 37 (Page 1 of 3). Edit Mask Patterns

Mask	Pattern	Examples	
		Value	Result
M0	IIIIIIIIIIITS	+01234	1234
		-00001	1-
M1	TTTTTTTTTTTTTTS	-00123	00123-
		+00123	00123
M2	I,III,III,III,IIT.TTS	+123450	1,234.50
		-000020	0.20-
M3	I,III,III,III,IIT.TTCR	-001234	12.34CR
		+123456	1,234.56
M4	SI,III,III,III,IIT.TT	+0123456	+1,234.56
		-1234567	-12,345.67

Table 37 (Page 2 of 3). Edit Mask Patterns

Mask	Pattern	Examples	
		Value	Result
M5	SI,III,III,III,IIT.TTS	-001234	(12.34)
		+123450	1,234.50
M6	III-TTT-TTTT	00123456	012-3456
		12345678	1-234-56788
M7	TTT-TT-TTTT	00123456	000-12-3456
		12345678	012-34-5678
M8	IT:TT:TT	030553	3:05:53
		121736	12:17:36
M9	IT/TT/TT	123094	12/30/94
		083194	8/31/94
M10	IIIIIIIIIIIT	01234	1234
		00000	0
M11	TTTTTTTTTTTTTTT	00010	00010
		01234	01234
M12	SIII,III,III,III,IIT	+1234567	1,234,567
		-0012345	-12,345
M13	SIII.III.III.III.IIT	+1234567	1.234.567
		-0012345	-12.345
M14	SIII III III III IITS	+1234567	1 234 567
		-0012345	(12 345)
M15	III III III III IITS	+1234567	1 234 567
		-0012345	12 345-
M16	SIII III III III IIT	+1234567	1 234 567
		-0012345	-12 345
M17	SIII'III'III'III'IIT	+1234567	1'234'567
		-0012345	-12'345
M18	SI,III,III,III,IIT.TT	+0123456	1,234.56
		-1234567	-12,345.67
M19	SI.III.III.III.IIT.TT	+0123456	1.234.56
		-1234567	-12.345.67
M20	SI III III III IIT,TTS	+0123456	1 234,56
		-1234567	(12 345,67)
M21	I III III III IIT,TTS	+0123456	1 234,567
		-1234567	12 345,67-
M22	SI III III III IIT,TT	+0123456	1 234,56
		-1234567	-12 345,67

Table 37 (Page 3 of 3). Edit Mask Patterns

Mask	Pattern	Examples	
		Value	Result
M23	S'IIII'IIII'IT.TT	+0123456	1'234.56
		-1234567	-12'345.67
M24	S'IIII'IIII'IT,TT	+0123456	1'234.56
		-1234567	-12'345.67
M25	SIIIIIIIIIIIT	+01234	1234
		-00001	-1

The elements used in the representation of the edit masks in Table 37 on page 46 are as follows:

- **I** indicates a leading insignificant digit. If zero, this digit will not be shown.
- **T** indicates a significant digit. If zero, this digit will be shown.
- **CR** (in M3) is printed to the right of the digits if the value is negative; otherwise, two blanks are printed to the right of the digits.
- **S** indicates a sign. If it appears as the first character in the pattern, it is a leading sign. If it appears as the last character in the pattern, it is a trailing sign. The examples in Table 37 on page 46 show the applicable positive and negative signs for each edit mask.
- Any other character (for example, /) will be printed as shown.

You can also use these elements to specify your own patterns in the EDIT subparameter. For example,

```
EDIT=(IIT.T)
```

Edit masks and edit patterns can be used for data fields (OUTREC parameter) and for statistics (TOTAL, SUBTOTAL, MAX, SUBMAX, MIN, SUBMIN, AVG, and SUBAVG subparameters) in trailers, as you will see in this example.

Another OUTREC feature used in this example is table lookup and change. This feature allows you to look up a field in your OUTFIL input records in a table of character, hexadecimal or bit constants, and change those that match to another specified character or hexadecimal constant. This means that you can substitute meaningful words or phrases for cryptic values (for example, READ can be substituted for R, EMPTY can be substituted for X'FF' or VSAM can be substituted for bit 1 = 0). In this example, the table lookup and change feature is used to substitute the full names of publishers for their abbreviations.

To give you an overview of how the report is specified, all of the JCL and control statements for this example are shown below. Note that the input consists of two data sets concatenated together and that we are using only the first 50 characters of the Book Title field.

```

//REPORT    JOB  A492,PROGRAMMER
//SORT      EXEC  PGM=SORT
//STEPLIB   DD  DSN=A492.SM,DISP=SHR
//SYSOUT    DD  SYSOUT=A
//SORTIN    DD  DSN=A123456.SORT.SAMPIN,DISP=SHR
//          DD  DSN=A123456.SORT.SAMPADD,DISP=SHR
//BOOKLIST  DD  SYSOUT=A
//SORTWK01  DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN     DD  *
SORT FIELDS=(1,50,CH,A)
OUTFIL FNames=BOOKLIST,LINES=40,
      HEADER2=(2:'BOOKSTORE LIST',
               50:'DATE: ',DATE,
               82:'PAGE: ',PAGE,
               3/,5:'BOOK TITLE',
               60:'PUBLISHER',
               85:'PRICE ($)',
               /,5:50'- ',
               60:15'- ',
               85:9'- '),
      OUTREC=(5:1,50,
               60:106,4,
               CHANGE=(15,
                       C'FERN',C'FERNALL BROS.',
                       C'COR',C'CORNISH LTD.',
                       C'VALD',C'VALDERN AND CO.',
                       C'WETH',C'WETHMAN INC.'),
               NOMATCH=(C'UNKNOWN'),
               80:170,4,BI,M18,
               133:X),
      TRAILER1=(2:'SUMMARY FOR ',DATE,
                82:'PAGE: ',PAGE,
                3/,2:'BOOKS IN STOCK: ',COUNT,
                3/,2:'AVERAGE PRICE OF BOOKS IN STOCK: ',
                AVG=(170,4,BI,EDIT=($I,IIT.TT)))
/*

```

The steps explaining the OUTFIL statement are divided into four sections to make them easier to understand.

Using the FNames and LINES Parameters

This section describes how to use the FNames parameter to identify the ddname of the OUTFIL data set and the LINES parameter to control the report page length.

Step	Action
1	Leave at least one blank and type OUTFIL .
2	Leave at least one blank and type FNames=BOOKLIST followed by a comma.
3	Type LINES= and the number of lines per page for the report, which is 40, followed by a comma.

```
OUTFIL FNames=BOOKLIST,LINES=40,
```

Using the HEADER2 Parameter

This section describes how to use the HEADER2 parameter to define the contents of the page header that will appear at the top of each page of the report, except for the summary sheet. For our example, the page header consists of several lines, including a title line identifying a title for the report and the date and page number, and appropriate underlined headings for the columns of data.

Step	Action
4	On the next line, leave at least one blank and type HEADER2=

Step	Action
5	<p>Type, in parentheses and separated by commas, the following:</p> <ol style="list-style-type: none"> 1. The column you want the first title string to start in, which is 2, followed by a colon. 2. The title string 'BOOKSTORE LIST' in single quotes. 3. On the next line, the column you want the second title string to start in, which is 50, followed by a colon. You may have to iteratively run the report job and adjust the column values until you get the report looking the way you want it to. 4. The title string 'DATE: ' in single quotes. 5. The subparameter DATE to request the current date. 6. On the next line, the column you want the third title string to start in, which is 82, followed by a colon. 7. The title string 'PAGE: ' in single quotes. 8. The subparameter PAGE to request the current page number. 9. On the next line, 3/ to insert two blank lines before the column headings. <i>n/</i> results in <i>n-1</i> blank lines in the report. 10. The column you want the first column heading to start in, which is 5, followed by a colon. 11. The column heading 'BOOK TITLE' in single quotes. 12. On the next line, the column you want the second column heading to start in, which is 60, followed by a colon. 13. The column heading 'PUBLISHER' in single quotes. 14. On the next line, the column you want the third column heading to start in, which is 85, followed by a colon. 15. The column heading 'PRICE (\$)' in single quotes. 16. On the next line, / to start a new line in the report. 17. The column you want the first column underlining to start in, which is 5 (the same as that used for the first column heading), followed by a colon. 18. The number of dashes needed to underline the first column heading, which is 50 from the length of the Book Title field, followed by - in single quotes. 19. On the next line, the column you want the second column underlining to start in, which is 60, followed by a colon. 20. The number of dashes needed to underline the second column heading, which is 15 from the length of the substituted Publisher names (see CHANGE below), followed by - in single quotes. 21. On the next line, the column you want the third column underlining to start in, which is 85, followed by a colon. 22. The number of dashes needed to underline the third column heading, which is 9 from the length of the third column heading, followed by - in single quotes. You may have to iteratively run the report job to determine the width of certain columns, especially for edited numbers.

Using OUTFIL

```
HEADER2=(2:'BOOKSTORE LIST',
          50:'DATE: ',DATE,
          82:'PAGE: ',PAGE,
          3/,5:'BOOK TITLE',
          60:'PUBLISHER',
          85:'PRICE ($)',
          /,5:50'- ',
          60:15'- ',
          85:9'- '),
```

Using the OUTREC Parameter

This section describes how to use the OUTREC parameter to arrange your selected input fields as columns of data for the report.

Step	Action
6	On the next line, leave at least one blank and type OUTREC=
7	<p>Type, in parentheses, and separated by commas, the following:</p> <ol style="list-style-type: none"> 1. The column you want the Book Title to start in, which is 5, followed by a colon. 2. The location (1) and length (50) of the Book Title field. Although the full Book Title field is 75 characters, we are only using the first 50 characters of the field here. 3. On the next line, the column you want the expanded Publisher field to start in, which is 60, followed by a colon. 4. The location (106) and length (4) of the Publisher field. 5. On the next line, the subparameter CHANGE= which describes your lookup and change table. OUTFIL's lookup and change feature can be used in many ways in output records and reports to substitute meaningful words and phrases for cryptic character, hexadecimal and bit values. 6. Type, in parentheses, and separated by commas, the following: <ol style="list-style-type: none"> a. The length for the expanded Publisher field, which is 15. b. On the next line, the lookup table to convert the Publisher field to the expanded Publisher field, which consists of a character constant for each expected input field value (for example, C'FERN') followed by a character constant for the associated output field value (for example, C'FERNALL BROS.'). 7. On the next line, the subparameter NOMATCH= which indicates the action to be taken for an unexpected input value. Use the NOMATCH feature to identify invalid or unexpected values in your report. If you don't specify NOMATCH, DFSORT issues a message and terminates for an unexpected value. 8. Type, in parentheses, the character constant for an unexpected input field value, which is C'UNKNOWN'. 9. On the next line, the column you want the edited Price field to start in, which is 80, followed by a colon. 10. The location (170), length (4), format (BI) and edit mask (M18) for the Price field. 11. On the next line, the column you want the report and data records to end in, which is 133, followed by a colon and X to indicate one blank. This ensures that your data records are at least as long as your report records.

```

OUTREC=(5:1,50,
        60:106,4,
        CHANGE=(15,
                C'FERN',C'FERNALL BROS.',
                C'COR',C'CORNISH LTD.',
                C'VALD',C'VALDERN AND CO.',
                C'WETH',C'WETHMAN INC.'),
        NOMATCH=(C'UNKNOWN'),
        80:170,4,BI,M18,
        133:X),

```

Using the TRAILER1 Parameter

This section describes how to use the TRAILER1 parameter to define the contents of the summary page.

Step	Action
8	On the next line, leave at least one blank and type TRAILER1=
9	Type, in parentheses and separated by commas, the following: <ol style="list-style-type: none"> 1. The column you want the summary title to start in, which is 2, followed by a colon. 2. The summary title 'SUMMARY FOR ' in single quotes. 3. The subparameter DATE to request the current date. 4. The column you want the page string to start in, which is 82, followed by a colon, the title string 'PAGE: ' in single quotes, and the subparameter PAGE to request the current page number. 5. On the next line, 3/ to insert two blank lines before the count line. 6. The column you want the count line to start in, which is 2, followed by a colon. 7. The count line title 'BOOKS IN STOCK: ' in single quotes. 8. The subparameter COUNT to request a count of the number of data records in the report. 9. On the next line, 3/ to insert two blank lines before the average line. 10. The column you want the average line to start in, which is 2, followed by a colon. 11. The average line title 'AVERAGE PRICE OF BOOKS IN STOCK: ' in single quotes. 12. On the next line, the subparameter AVG= to request the average of the book prices. 13. Type, in parentheses and separated by commas, the location (170), length (4), format (BI) and edit pattern (EDIT=(\$I,IIT.TT)) for the Price field.

```
TRAILER1=(2:'SUMMARY FOR ',DATE,
          82:'PAGE: ',PAGE,
          3/,2:'BOOKS IN STOCK: ',COUNT,
          3/,2:'AVERAGE PRICE OF BOOKS IN STOCK: ',
          AVG=(170,4,BI,EDIT=($I,IIT.TT)))
```

This example creates the following three-page report:

BOOKSTORE LIST

DATE: 03/07/95

PAGE: 1

BOOK TITLE	PUBLISHER	PRICE (\$)
A SMALLER WORLD: MICROBES	FERNALL BROS.	19.95
ADVANCED TOPICS IN PSYCHOANALYSIS	FERNALL BROS.	26.00
ANOTHER ITALIAN DICTIONARY	CORNISH LTD.	9.25
ANTICIPATING THE MARKET	WETHMAN INC.	20.00
CELLS AND HOW THEY WORK	VALDERN AND CO.	24.95
CIVILIZATION SINCE ROME FELL	WETHMAN INC.	13.50
COMPLETE SPANISH DICTIONARY	VALDERN AND CO.	6.50
COMPUTER LANGUAGES	FERNALL BROS.	26.00
COMPUTERS: AN INTRODUCTION	WETHMAN INC.	18.99
CRISES OF THE MIDDLE AGES	CORNISH LTD.	12.00
DESIGNING APPLICATIONS	CORNISH LTD.	14.35
DNA: BLUEPRINT FOR YOU	FERNALL BROS.	21.95
EDITING SOFTWARE MANUALS	VALDERN AND CO.	14.50
EIGHTEENTH CENTURY EUROPE	WETHMAN INC.	17.90
FRENCH TO ENGLISH DICTIONARY	FERNALL BROS.	11.00
FREUD'S THEORIES	VALDERN AND CO.	12.50
GUIDE TO COLLEGE LIFE	WETHMAN INC.	20.00
GUNTHER'S GERMAN DICTIONARY	WETHMAN INC.	10.88
INKLINGS: AN ANTHOLOGY OF YOUNG POETS	CORNISH LTD.	5.95
INTRODUCTION TO BIOLOGY	VALDERN AND CO.	23.50
INTRODUCTION TO PSYCHOLOGY	CORNISH LTD.	22.00
KNOW YOUR CONSUMER	CORNISH LTD.	45.00
LIVING WELL ON A SMALL BUDGET	CORNISH LTD.	99.00
MAP OF THE HUMAN BRAIN	CORNISH LTD.	8.95
MODERN ANTHOLOGY OF WOMEN POETS	CORNISH LTD.	4.50
NOVEL IDEAS	VALDERN AND CO.	24.50
NUMBERING SYSTEMS	FERNALL BROS.	3.60
PICK'S POCKET DICTIONARY	CORNISH LTD.	2.95
POLITICS AND HISTORY	FERNALL BROS.	9.95
QUEUE THEORY	FERNALL BROS.	15.00
REBIRTH FROM ITALY	WETHMAN INC.	25.60
SHORT STORIES AND TALL TALES	VALDERN AND CO.	15.20
STRATEGIC MARKETING	VALDERN AND CO.	23.50
SUPPLYING THE DEMAND	CORNISH LTD.	19.25
SYSTEM PROGRAMMING	WETHMAN INC.	31.95

BOOKSTORE LIST

DATE:03/07/95

PAGE: 2

BOOK TITLE	PUBLISHER	PRICE (\$)
THE ANIMAL KINGDOM	CORNISH LTD.	30.00
THE ART OF TAKEOVERS	FERNALL BROS.	6.15
THE COMPLETE PROOFREADER	FERNALL BROS.	6.25
THE INDUSTRIAL REVOLUTION	WETHMAN INC.	7.95
THE TOY STORE TEST	CORNISH LTD.	26.00
VIDEO GAME DESIGN	VALDERN AND CO.	21.99
ZEN BUSINESS	VALDERN AND CO.	12.00

SUMMARY FOR 03/07/95

PAGE: 3

BOOKS IN STOCK: 42

AVERAGE PRICE OF BOOKS IN STOCK: \$18.83

Summary

This chapter covered how to use OUTFIL to:

- Create multiple copies
- Create multiple output data sets with unique content
- Create reports

The next chapter will cover methods of calling DFSORT from a program.

Chapter 7. Calling DFSORT from a Program

This chapter contains Programming Interface information.

In addition to processing your DFSORT program control statements with a JCL EXEC statement, you can call DFSORT from programs written in COBOL, PL/I, or assembler language. In this chapter, you will concentrate on sorting and merging using COBOL and sorting using PL/I. The examples in this chapter assume that the COBOL environment is available.

For information on restrictions when using these languages and on calling DFSORT from an assembler program, see *Application Programming Guide*.

Passing Control Statements

When using COBOL, or PL/I, you can pass the INCLUDE, OMIT, SUM, INREC, OUTREC, and OUTFIL control statements to DFSORT by using the SORTCNTL DD or DFSPARM DD statement. These program products create a RECORD control statement and a SORT or MERGE control statement for you. For example, you can use the SORTCNTL DD statement to pass the INCLUDE control statement that selects only the English department books:

```
//EXAMP    JOB A492,PROGRAMMER
.
.
.
//SORTCNTL DD *
           INCLUDE COND=(110,5,CH,EQ,C'ENGL')
/*
```

The SYSIN DD statement is not used for passing control statements. Instead, control statements are passed by the SORTCNTL DD or DFSPARM DD statement.

When using VS COBOL II, COBOL for MVS & VM or COBOL for OS/390 & VM, you need to understand the use of the SORT-CONTROL and SORT-RETURN special registers. For full information, see the COBOL Programmer's Guide that describes the compiler version available at your site.

Calling DFSORT from a COBOL Program

To call DFSORT from a COBOL program, use the COBOL statements SORT and MERGE. This section shows sample programs that use the COBOL SORT and MERGE statements. For complete information, see the COBOL Programmer's Guide describing the compiler version available at your site.

Sorting Records

The sample COBOL program in Figure 2 on page 59 calls DFSORT to sort the bookstore master file (MASTER-FILE) by title in ascending order. The sorted master file is written to SORTED-MASTER-FILE.

Following is the JCL that calls the sample COBOL program:

Calling DFSORT from a Program

```
//EXAMP    JOB   A492,PROGRAMMER
//BOOKS    EXEC  PGM=COBOLPGM
//STEPLIB  DD    DSN=A492.SM,DISP=SHR
//          DD    DSN=USER.PGMLIB,DISP=SHR
//SYSOUT   DD    SYSOUT=A
//MASTIN   DD    DSN=A123456.MASTER,DISP=OLD
//SORTWK01 DD    UNIT=SYSDA,SPACE=(CYL,(1,1))
//MASTOUT  DD    DSN=A123456.OUTB,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1)),UNIT=SYSDA
//PRINTFL  DD    SYSOUT=A
/*
```

In contrast to the JCL for executing DFSORT with the JCL EXEC statement (see “Sorting Data Sets with the JCL EXEC Statement” on page 15) the above JCL has these differences:

- The program name on the EXEC statement is that of the COBOL program.
- The STEPLIB DD statement defines the library containing the DFSORT program, as well as the library containing the COBOL program.
- The name of the DD statement for the input file need not be SORTIN.
- The name of the DD statement for the output file need not be SORTOUT.

Notice that the control field and order of the sort are specified in the COBOL program itself rather than with a SORT control statement. Figure 2 on page 59 shows the sample COBOL program.

```

IDENTIFICATION DIVISION.
PROGRAM-ID.
    COBOLPGM.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SD-FILE ASSIGN TO
        DUMMYNM.
    SELECT MASTER-FILE ASSIGN TO
        MASTIN.
    SELECT SORTED-MASTER-FILE ASSIGN TO
        MASTOUT.
    SELECT PRINT-FILE ASSIGN TO
        PRINTFL.
DATA DIVISION.
FILE SECTION.
SD  SD-FILE
    DATA RECORD IS SD-RECORD.
01  SD-RECORD.
    05 TITLE-IN    PICTURE X(75).
    05 AUTH-LN-IN  PICTURE X(15).
    05 AUTH-FN-IN  PICTURE X(15).
    05 PUB-IN      PICTURE X(4).
    05 COUR-DEPT-IN PICTURE X(5).
    05 COUR-NO-IN  PICTURE X(5).
    05 COUR-NAM-IN PICTURE X(25).
    05 INST-LN-IN  PICTURE X(15).
    05 INST-INIT-IN PICTURE X(2).
    05 NO-STOCK-IN PICTURE 9(8) COMP.
    05 NO-SOLD-IN  PICTURE 9(8) COMP.
    05 PRICE-IN    PICTURE 9(8) COMP.

```

Figure 2 (Part 1 of 2). Sample COBOL Program with SORT Commands

```
FD  MASTER-FILE
   DATA RECORD IS MASTER-RECORD.
01  MASTER-RECORD.
   05 FILLER      PICTURE X(173).

FD  SORTED-MASTER-FILE
   DATA RECORD IS SORTED-MASTER-RECORD.
01  SORTED-MASTER-RECORD.
   05 FILLER      PICTURE X(173).

FD  PRINT-FILE
   DATA RECORD IS OUTPUT-REPORT-RECORD.
01  OUTPUT-REPORT-RECORD.
   05 REPORT-OUT  PICTURE X(120).
.
.
.

PROCEDURE DIVISION.
.
.
.

SORT-ROUTINE SECTION.
  SORT SD-FILE
  ASCENDING KEY TITLE-IN
  USING MASTER-FILE
  GIVING SORTED-MASTER-FILE.
  IF SORT-RETURN > 0
  DISPLAY "SORT FAILED".
.
.
.

SORT-REPORT SECTION.
  print a report on PRINT-FILE using SORTED-MASTER-FILE.
.
.
.
  STOP RUN.
```

Figure 2 (Part 2 of 2). Sample COBOL Program with SORT Commands

Merging Records

The sample COBOL program in Figure 3 on page 62 calls DFSORT to merge the presorted bookstore master file (MASTER-FILE) with another presorted file (NEW-BOOKS-FILE) to create a new master file (MERGED-FILE).

The JCL for the program is as follows:

```
//EXAMP    JOB  A492,PROGRAMMER
//BOOKS    EXEC PGM=COBOLP
//STEPLIB  DD   DSN=A492.SM,DISP=SHR
//          DD   DSN=USER.PGMLIB,DISP=SHR
//SYSOUT    DD   SYSOUT=A
//MASTERFL DD   DSN=A123456.MASTER,DISP=OLD
//NEWBOOKS DD   DSN=A123456.NEW,DISP=OLD
//MERGEDFL DD   DSN=A123456.OUTC,DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(1,1)),UNIT=SYSDA
/*
```

Figure 3 on page 62 shows the sample COBOL program.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID.  
    COBOLP.  
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
    SELECT SD-FILE ASSIGN TO  
        DUMMYNM.  
    SELECT MASTER-FILE ASSIGN TO  
        MASTERFL.  
    SELECT NEW-BOOKS-FILE ASSIGN TO  
        NEWBOOKS.  
    SELECT MERGED-FILE ASSIGN TO  
        MERGEDFL.  
DATA DIVISION.  
FILE SECTION.  
SD  SD-FILE  
    DATA RECORD IS SD-RECORD.  
01  SD-RECORD.  
    05 TITLE-KEY  PICTURE X(75).  
    05 FILLER     PICTURE X(98).  
  
FD  MASTER-FILE  
    DATA RECORD IS MASTER-RECORD.  
01  MASTER-RECORD.  
    05 FILLER     PICTURE X(173).  
  
FD  NEW-BOOKS-FILE  
    DATA RECORD IS NEW-BOOKS-RECORD.  
01  NEW-BOOKS-RECORD.  
    05 FILLER     PICTURE X(173).  
  
FD  MERGED-FILE  
    DATA RECORD IS MERGED-RECORD.  
01  MERGED-RECORD.  
    05 FILLER     PICTURE X(173).  
.  
.  
.  
  
PROCEDURE DIVISION.  
.  
.  
.  
  
MERGE-ROUTINE SECTION.  
    MERGE SD-FILE  
    ASCENDING KEY TITLE-KEY  
    USING MASTER-FILE NEW-BOOKS-FILE  
    GIVING MERGED-FILE.  
    IF SORT-RETURN > 0  
    DISPLAY "MERGE FAILED".  
    STOP RUN.
```

Figure 3. Sample COBOL Program with MERGE Commands

Sorting with COBOL FASTSRT

If you compile the COBOL program in Figure 3 on page 62 for sorting records with VS COBOL II, COBOL for MVS & VM, or COBOL for OS/390 & VM, the input (from MASTER-FILE) and the output (to SORTED-MASTER-FILE) would qualify for the COBOL FASTSRT option. With this compile-time FASTSRT option, your sort runs considerably faster, because DFSORT rather than COBOL does the input and output processing. For full information on FASTSRT, refer to *Application Programming Guide* and the COBOL Programmer's Guide that describes the compiler version available at your site.

Note: COBOL evaluates sort input and output independently to see if it qualifies for FASTSRT. If either the input or the output of your sort does not qualify because of the presence of an input or output procedure, you might be able to replace such a procedure and use DFSORT control statements to accomplish the same thing. For example, you can use a control statement (OUTREC) to indicate how records will be reformatted before being written to the output data set.

Calling DFSORT from a PL/I Program

When calling DFSORT, a PL/I program must pass a SORT control statement, a RECORD control statement, and the amount of main storage to be allocated to DFSORT. On the RECORD control statement, you specify the record type and length. Following the RECORD control statement, you specify the amount of main storage in bytes. (DFSORT performs best when the main storage value is at least 1 megabyte.)

You can also pass control statements by using the SORTCNTL DD statement.

This JCL is for the program shown in Figure 4 on page 64. The SORTCNTL DD statement is used to pass an INCLUDE control statement that selects only the English department books.

```
//EXAMP    JOB   A492,PROGRAMMER
//BOOKS    EXEC  PGM=PLIPGM
//STEPLIB  DD    DSN=A492.SM,DISP=SHR
//          DD    DSN=USER.PGMLIB,DISP=SHR
//SYSOUT   DD    SYSOUT=A
//SORTIN   DD    DSN=A123456.SORT.SAMPIN,DISP=SHR
//SORTWK01 DD    UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTOUT  DD    DSN=A123456.SORT.SAMPOUT,DISP=OLD
//          SPACE=(CYL,(1,1)),UNIT=SYSDA
//SORTCNTL DD *
//          INCLUDE COND=(110,5,CH,EQ,C'ENGL')
//SYSPRINT DD    SYSOUT=A
/*
```

The sample PL/I program shown in Figure 4 on page 64 calls DFSORT to sort the bookstore file by title. It allocates 2 000 000 bytes of main storage.

```
PLIPGM:  PROC OPTIONS(MAIN);

          DCL 1 MASTER_RECORD,
            5 TITLE_IN   CHAR(75),
            5 AUTH_LN_IN  CHAR(20),
            .
            .
            .
            5 PRICE_IN   BIN FIXED(31);

          DCL RETURN_CODE FIXED BIN(31,0);
          .
          .
          .
          CALL PLISRTA (' SORT FIELDS=(1,75,CH,A) ',
                       ' RECORD TYPE=F,LENGTH=(173) ',
                       20000000,
                       RETURN_CODE);
          IF RETURN_CODE  $\neq$  0 THEN DO;
              PUT SKIP EDIT ('SORT FAILED')(A);
              CALL PLIRETC(RETURN_CODE);
          END;
          .
          .
          .
          CALL OUTPUT;
          .
          .
          .
          OUTPUT: PROCEDURE;
          .
          .
          .
          . Print a report from the sorted master file (SORTOUT)
          .
          .
          .
          END;
END PLIPGM;
```

Figure 4. Sample PL/I Program with SORT Commands

Summary

This chapter covered methods of calling DFSORT from COBOL, and PL/I.

Chapter 8. Overriding Installation Defaults

IBM provides DFSORT with pre-set defaults. During installation, your system programmer can change these defaults. For example, one IBM-supplied default is to list the DFSORT control statements in the output data set for messages. However, at your site, the default might be to *not* list the control statements.

Furthermore, you can establish separate defaults for jobs executed with JCL and those called from a program. So, when you execute DFSORT with JCL, the default might be to list the statements, and when you call DFSORT from a program, the default might be to *not* list them.

You can temporarily override some of the installation defaults either by specifying parameters on the JCL EXEC statement or by writing an OPTION control statement. When calling DFSORT from an Assembler program, you can also override defaults by using a parameter list.

In this chapter, you will learn how to override some of the many available defaults, concentrating on the JCL EXEC statement and the OPTION control statement. For a list of all the possible defaults, and information on how to code the assembler parameter list, see *Application Programming Guide*. See also Chapter 10, "Using the ICETOOL Utility" on page 73 on using the ICETOOL DEFAULTS parameter to print a list of your installation defaults.

Specifying PARM Parameters on a JCL EXEC Statement

While executing DFSORT with the JCL EXEC statement, you can use the PARM parameter to override certain defaults. For example, if the default at your site is to list DFSORT statements and you do not want them listed, you can specify NOLIST in the PARM field:

```
//SORT EXEC PGM=SORT,PARM='NOLIST'
```

On the other hand, if the default is not to list the statements and you do want them listed, you can specify LIST in the PARM field:

```
//SORT EXEC PGM=SORT,PARM='LIST'
```

Writing an OPTION Control Statement

Whether you execute DFSORT with the JCL EXEC statement or call it from a program, you can use the OPTION statement to override certain defaults. To do so, place the OPTION statement among the other DFSORT control statements that follow the SYSIN or SORTCNTL DD statement.

A particular default that can be overridden with the OPTION statement is one that specifies whether equally collating records are to be written in their original order.

The IBM default is for DFSORT to write equally collating records in random order. If your site has kept this default and you want to temporarily override it (so that

Overriding Installation Defaults

equally collating records are written in their *original* order), you can specify EQUALS on the OPTION statement:

```
OPTION  EQUALS
```

Or, if your site has established EQUALS as the default and you want to temporarily override it (so that equally collating records are written in *random* order), you can specify NOEQUALS on the OPTION statement:

```
OPTION  NOEQUALS
```

Note that some defaults can be overridden by the OPTION control statement, but not the JCL EXEC statement, and that other defaults can be overridden by the JCL EXEC statement, but not the OPTION control statement.

For a table showing all defaults and exactly how each can be overridden, see *Application Programming Guide*.

Specifying DFSPARM Parameters

PARM options and DFSORT control statements can also be specified with the DFSPARM DD statement to override installation options. See *Application Programming Guide* for more information.

Summary

This chapter covered methods of overriding the installation defaults using a JCL EXEC statement or an OPTION control statement.

Chapter 9. Using DFSORT Efficiently

You will get the best performance from DFSORT if you follow these guidelines:

- Be generous with main storage.
- Use high-speed disks or Hiperspace for work space.
- Eliminate unnecessary fields with INREC.
- Eliminate unnecessary records with INCLUDE or OMIT.
- Reduce file size with STOPAFT and SKIPREC.
- Consolidate records with SUM.
- Create multiple output data sets with OUTFIL
- Run DFSORT with the JCL EXEC statement.
- Use FASTSORT with COBOL.
- Avoid options that might degrade performance.

Additional suggestions can be found in *Application Programming Guide and Tuning Guide*.

Be Generous with Main Storage

In general, the more main storage available to DFSORT, the better the performance. This is especially true when the input data set is larger than available main storage.

When DFSORT was installed, your system programmer selected a default value for main storage to be used for all jobs at your site. The main storage value is reported as the SIZE value in the DFSORT messages. If the default storage value is less than about 1 megabyte (1024K), you might want to make it larger when you sort large data sets.

You can use SIZE=nM in the PARM field of the JCL EXEC statement to change the storage value. For example:

```
//SORT EXEC PGM=SORT,PARM='SIZE=2M'
```

Alternatively, you can use MAINSIZE=nM on the OPTION statement to change the storage value. For example:

```
OPTION MAINSIZE=2M
```

Use High-Speed DASD or Hiperspace

Using high-speed DASD, such as the IBM 3390, or Hiperspace (through DFSORT's Hipersorting capability) for work space offers the best performance. You should avoid using tapes for work space whenever possible.

Eliminate Unnecessary Fields with INREC

If you need to reformat your records, using INREC to significantly shorten them can result in faster processing.

Remember that INREC reformats records *before* they are processed, and OUTREC reformats them *after* they are processed. Therefore, you should use INREC to shorten records and OUTREC to lengthen records. For a summary of the control statements and the corresponding record positions to refer to when using INREC, see Table 38.

Table 38. Control Statement and Corresponding Records with INREC

Control Statement	Original Records	Reformatted Records
SORT		✓
MERGE		✓
SUM		✓
OUTREC		✓
OUTFIL		✓
INCLUDE	✓	
OMIT	✓	

Eliminate Unnecessary Records with INCLUDE or OMIT

Naturally, the size of the input file(s) also affects the amount of time processing will take. The fewer the records, the faster the DFSORT application. You can improve performance by using INCLUDE or OMIT whenever possible to select only the records pertaining to your application.

Reduce File Size with STOPAFT and SKIPREC

You can also use the STOPAFT and SKIPREC options to reduce the size of the input file.

- Use STOPAFT to specify the maximum number of records that should be accepted for sorting or copying.
- Use SKIPREC to skip a specified number of records at the beginning of the input file being sorted or copied.

For information on how to use these options, see *Application Programming Guide*.

Consolidate Records with SUM

You can improve performance by using the SUM statement, if appropriate for your job, to either:

- Add the contents of fields whenever two records with equal control fields are found. DFSORT places the result in one record and deletes the other, reducing the number of records to be sorted or merged.
- Delete records with duplicate control fields by specifying FIELDS=NONE in a SUM statement.

For details on these methods, see Chapter 4, “Summing Records” on page 27.

Create Multiple Output Data Sets with OUTFIL

If you need to create multiple output data sets from the same input data set, you can use OUTFIL to read the input data set only once, thus improving performance. OUTFIL can be used for sort, merge, and copy applications to provide sophisticated filtering, editing, conversion, lookup and replace, and report features.

Run DFSORT with JCL

As a rule, DFSORT is more efficient when executed with the JCL EXEC statement than when called from a program.

Although calling DFSORT from a program might be convenient if the program modifies the data sets before or after DFSORT (for example, if DFSORT sums numbers and the program calculates their average), be aware of the possible trade-off in performance.

Use FASTSRT with COBOL

With VS COBOL II, COBOL for MVS & VM and COBOL for OS/390 & VM, using the FASTSRT compile-time option enhances DFSORT performance. With FASTSRT, DFSORT rather than COBOL does the input and output processing. For more information on this option, see the COBOL Programmer's Guide that describes the compiler version available at your site.

Avoid Options That Might Degrade Performance

The options listed below might adversely affect DFSORT performance. Use them only when necessary. For a full description of what these options are and how they affect performance, see *Application Programming Guide* and *Tuning Guide*.

- VERIFY option
- EQUALS option
- NOBLKSET option
- CKPT option
- EQUCOUNT option
- NOCINV option
- LOCALE option
- Tape work data sets
- User exit routines
- EFS program
- Dynamic link-edit of user exit routines
- Certain DEBUG options (BSAM, NOASSIST, NOCFW)
- Small values for HIPRMAX, DSPSIZE, or MAINSIZE options

Summary

This chapter covered methods of improving DFSORT performance, including the effective use of control statements and options.

Part 3. Learning to Use ICETOOL

Chapter 10. Using the ICETOOL Utility	73
ICETOOL Operators	73
Input Data Sets	74
Creating an ICETOOL Job	75
Writing Required JCL Statements	75
ICETOOL Comment and Blank Statements	76
Printing Statistics For Numeric Fields	77
Continuing an Operator Statement	78
Statistics For Record Lengths	79
Creating Identical Sorted Data Sets	79
Creating Different Subsets of a Sorted Data Set	82
Creating Multiple Unsorted Data Sets	84
Counting Values in a Range	85
Printing Simple Reports	87
Printing Tailored Reports	88
Using Formatting Items	90
Edit Masks	90
Division	92
Leading, Floating and Trailing Characters	92
Printing Sectioned Reports	93
Printing How Many Times Fields Occur	96
Selecting Records by Field Occurrences	97
Complete ICETOOL Job and TOOLMSG Output	98

Chapter 10. Using the ICETOOL Utility

ICETOOL is a multipurpose DFSORT utility that uses the capabilities of DFSORT to perform multiple operations on one or more data sets in a single step.

This chapter will show you how to write an ICETOOL job that uses several of the ICETOOL operators. To fully use the capabilities of ICETOOL, you should become familiar with all of its operators, operands, and methods of invocation, as described in *Application Programming Guide* and its messages and operator return codes as described in *Messages, Codes and Diagnosis*.

ICETOOL Operators

The twelve ICETOOL operators listed below can be used to perform a variety of functions. By using various combinations of the twelve ICETOOL operators, you can easily create applications that perform many complex tasks.

COPY Copies a data set to one or more output data sets.

COUNT Prints a message containing the count of records in a data set.

DEFAULTS Prints the DFSORT installation defaults in a separate list data set.

DISPLAY Prints the values or characters of specified numeric or character fields in a separate list data set. Simple, tailored, or sectioned reports can be produced.

MODE Three modes are available which can be set or reset for groups of operators:

- STOP mode (the default) stops subsequent operations if an error is detected.
- CONTINUE mode continues with subsequent operations if an error is detected.
- SCAN mode allows ICETOOL statement checking without actually performing any operations.

OCCUR Prints each unique value for specified numeric or character fields and how many times it occurs in a separate list data set. Simple or tailored reports can be produced. The values printed can be limited to those for which the value count meets specified criteria (for example, only duplicate values or only non-duplicate values).

RANGE Prints a message containing the count of values in a specified range for a specified numeric field in a data set.

SELECT Selects records from a data set for inclusion in an output data set based on meeting criteria for the number of times specified numeric or character field values occur (for example, only duplicate values or only non-duplicate values). Records that are not selected can be saved in a separate output data set.

SORT Sorts a data set to one or more output data sets.

- STATS** Prints messages containing the minimum, maximum, average, and total for specified numeric fields in a data set.
- UNIQUE** Prints a message containing the count of unique values for a specified numeric or character field.
- VERIFY** Examines specified decimal fields in a data set and prints a message identifying each invalid value found for each field.

Input Data Sets

Each ICETOOL operator (except DEFAULTS and MODE) requires an input data set. The input data set used by one operator can be the same or different from the input data set used by another operator. Thus, ICETOOL can process many data sets in a single step.

This chapter uses as input data sets the branch office data set named SORT.BRANCH, the sample bookstore data set named SORT.SAMPIN, and the additional bookstore data set named SORT.SAMPADD. See “Creating Your Sample Input and Output Data Sets” on page 8 for additional information about these sample data sets. Two temporary data sets created by ICETOOL from SORT.BRANCH are also used as input.

Note: This chapter also uses data sets other than SORT.BRANCH, SORT.SAMPIN and SORT.SAMPADD. You can either create data sets from scratch to match the ones used in the text, or else perform a similar exercise on data sets you already have.

Table 39 shows the length and format of the fields in the branch office data set (SORT.BRANCH).

Table 39. Field Lengths and Formats for SORT.BRANCH

Field	Length	Data Format
City	15	CH
State	2	CH
Employees	4	ZD
Revenue	6	PD
Profit	6	PD

Table 40 on page 75 shows the records in the branch office data set.

Table 40. Branch Office Data Set Records

City	State	Employees	Revenue	Profit
1 15	16 17	18 21	22 27	28 33
Los Angeles	CA	32	22530	-4278
San Francisco	CA	35	42820	6832
Fort Collins	CO	22	12300	-2863
Sacramento	CA	29	42726	8276
Sunnyvale	CA	18	16152	-978
Denver	CO	33	31876	6288
Boulder	CO	32	33866	7351
Morgan Hill	CA	15	18200	3271
Vail	CO	19	23202	5027
San Jose	CA	21	27225	8264
San Diego	CA	22	32940	8275
Aspen	CO	20	25800	5200

Appendix B, “The Sample Bookstore Data Sets” on page 117 shows the length, format and contents of the fields in the records of the bookstore data sets (SORT.SAMPIN and SORT.SAMPADD).

Creating an ICETOOL Job

An ICETOOL job consists of:

1. The JCL statements that are required for every ICETOOL job.
2. The operator statements indicating the operations to be performed by the ICETOOL job.
3. The JCL statements that are required as a result of the specified operator statements.

Writing Required JCL Statements

The first step in creating any ICETOOL job is to write the JCL that is always required. Here is the required JCL for the job in this chapter:

```
//EXAMP JOB A492,PROGRAMMER
//TOOL EXEC PGM=ICETOOL,REGION=1024K
//STEPLIB DD DSN=A492.SM,DISP=SHR
//TOOLMSG DD SYSOUT=A
//DFSMSG DD SYSOUT=A
//TOOLIN DD *
<ICETOOL statements go here>
/*
<Additional JCL statements go here>
```

- The JOB statement signals the beginning of the job.
- The EXEC statement signals the beginning of the job step and tells the operating system to run the ICETOOL program. REGION=1024K is recommended.
- The STEPLIB statement defines the library containing DFSORT and ICETOOL (generally the same library). If these programs are in a system library, you can omit the STEPLIB statement.

- The TOOLMSG statement defines the output data set for ICETOOL messages.
- The DFSMSG statement defines the output data set for DFSORT messages.
- The TOOLIN statement precedes the ICETOOL statements (comment, blank, and operator statements). The ICETOOL statements you write must appear after TOOLIN. The additional JCL statements you write can appear before the TOOLIN statement or after the ICETOOL statements. In this ICETOOL job, the additional JCL statements will be placed after the ICETOOL statements.

ICETOOL Comment and Blank Statements

Comment statements and blank statements can be placed anywhere among the ICETOOL operator statements.

- Comment statements start with an asterisk (*) in column 1 and are printed along with the ICETOOL operator statements.
- Blank statements contain blanks in columns 1-72 and are ignored since ICETOOL prints blank lines where appropriate.

To write a blank statement and a comment statement for our example:

Table 41. Steps to Create a Blank Statement and a Comment Statement

Step	Action
1	After the TOOLIN DD statement, skip one line.
2	Type an asterisk (*) in column 1 followed by the comment.

When complete, the TOOLIN statements look like this:

```
//TOOLIN DD *

* Statistics from all branches
/*
```

For this ICETOOL job, a comment statement will be placed before each operator to describe its function. Although not required, this is a good practice to follow.

So Far

So far, you have been introduced to the basics of the ICETOOL utility. Now, using the following tutorials, you can learn about some of the operators. Each of the following sections contains a part of the same ICETOOL job, so that by the end of the chapter, you will have created a complete ICETOOL job. At the end of the chapter, there is a section that contains the complete job and its resulting messages.

Printing Statistics For Numeric Fields

When working with data sets containing numeric fields, you may want statistical information about one or more of those fields. You can use the STATS operator to find the minimum, maximum, average, and total values of up to 10 specific numeric fields.

To write a STATS statement that prints statistics for the employees, profit, and revenue fields of the branch office data set:

Table 42. Steps to Create the STATS Operator

Step	Action
1	Type STATS after the comment statement (you can leave one or more blanks before STATS if you like).
2	Leave at least one blank and type FROM(ALL) FROM specifies the ddname (that is, the name of the DD statement) for the input data set from which you want to print statistics. In this case ALL is the ddname chosen, but you can use any valid 1-8 character ddname you like.
3	Leave at least one blank and type ON ON defines a field for which you want to print statistics.
4	Type in parentheses, and separated by commas: <ol style="list-style-type: none">Where the employees field begins relative to the beginning of the input record (the first position is byte 1). The employees field begins at byte 18.The length of the employees field in bytes. The employees field is 4 bytes long.A code for the data format. The employees field contains zoned decimal data which you specify as ZD.
5	Leave at least one blank and type ON ON defines another field for which you want to print statistics. You can print statistics for up to 10 fields with one STATS statement. Specify the ON fields in the same order in which you want their statistics to be printed.
6	Type in parentheses, and separated by commas the location (28), length (6), and format (PD for packed decimal) of the profit field.
7	Leave at least one blank and type ON . Type in parentheses and separated by commas, the location (22), length (6), and format (PD) of the revenue field. Make sure that the statement is coded between columns 1 and 72.

When complete, the STATS operator statement looks like:

```
STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
```

Revenue

Profit

Employees

ddname of input data set

You must also write a DD statement for the A123456.SORT.BRANCH data set using the ddname ALL and place it at the end of the job:

```
//ALL DD DSN=A123456.SORT.BRANCH,DISP=SHR
```

When complete the TOOLIN statements and ALL statement look like:

```
//TOOLIN DD *  
  
* Statistics from all branches  
STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)  
/*  
//ALL DD DSN=A123456.SORT.BRANCH,DISP=SHR
```

When this STATS operation is run, the results are placed in the TOOLMSG data set. If you ran the ICETOOL job you created so far, the TOOLMSG output would look like:

```
ICE600I 0 DFSORT ICETOOL UTILITY RUN STARTED  
  
ICE632I 0 SOURCE FOR ICETOOL STATEMENTS: TOOLIN  
  
ICE630I 0 MODE IN EFFECT: STOP  
  
          * Statistics from all branches  
          STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)  
ICE627I 0 DFSORT CALL 0001 FOR COPY FROM ALL      TO E35 EXIT COMPLETED  
ICE628I 0 RECORD COUNT: 000000000000012  
ICE607I 0 STATISTICS FOR (18,4,ZD)      :  
ICE608I 0 MINIMUM: +000000000000015, MAXIMUM: +000000000000035  
ICE609I 0 AVERAGE: +000000000000024, TOTAL : +000000000000298  
ICE607I 0 STATISTICS FOR (28,6,PD)      :  
ICE608I 0 MINIMUM: -000000000004278, MAXIMUM: +000000000008276  
ICE609I 0 AVERAGE: +000000000004222, TOTAL : +0000000000050665  
ICE607I 0 STATISTICS FOR (22,6,PD)      :  
ICE608I 0 MINIMUM: +000000000012300, MAXIMUM: +000000000042820  
ICE609I 0 AVERAGE: +000000000027469, TOTAL : +0000000000329637  
ICE602I 0 OPERATION RETURN CODE: 00  
  
ICE601I 0 DFSORT ICETOOL UTILITY RUN ENDED - RETURN CODE: 00
```

Looking at the output, you will notice that:

- Message ICE628I gives the count of records processed.
- Messages ICE607I, ICE608I, and ICE609I give the numerical statistics for each ON field specified in the order in which they were specified.
- A return code for each operator is given in message ICE602I and the highest operator return code is given in message ICE601I.

Continuing an Operator Statement

If you cannot fit your STATS statement (or any other ICETOOL operator statement) between columns 1 and 72 of a single line, you can continue it across multiple lines. If you end a line with a dash (-) after the operator or any operand, the next line is treated as a continuation. Any characters specified after the dash are ignored.

Note that the operator and each operand must be completely specified on one line (between columns 1 and 72).

For example:

STATS - this is the operator
FROM(ALL) - ALL is the ddname for
 SORT.BRANCH
 ON(18,4,ZD) -
 ON(28,6,PD) -
 ON(22,6,PD)

Statistics For Record Lengths

When working with variable length record data sets, you can use the STATS operator to easily obtain the following information:

- The shortest record in the data set (minimum)
- The longest record in the data set (maximum)
- The average length of records in the data set (average)
- The total number of bytes in the data set (total)

ICETOOL provides the special ON(VLEN) field for printing these statistics. Specify ON(VLEN) as you would any other ON field.

So far

Now you know how to print statistics for numeric fields and record lengths. In the next section, you will learn how to work with the ICETOOL SORT operator.

Creating Identical Sorted Data Sets

You can use ICETOOL's SORT operator to create sorted output data sets. A single SORT operator can be used to create one output data set or up to 10 identical output data sets. Using INCLUDE or OMIT statements, you can select a subset of the input records. Using INREC or OUTREC statements, you can rearrange the fields of the input records. Using OUTFIL statements, you can create any number of output data sets with different subsets of records or arrangements of fields.

For this example, we will use both the sample bookstore data set (SORT.SAMPIN) and the additional bookstore data set (SORT.SAMPADD) as input. To write a SORT operator that selects the books from publishers VALD and WETH, sorts them by publisher and title, and writes them to DASD and print data sets:

Table 43 (Page 1 of 2). Steps to Create the SORT Operator

Step	Action
1	Write a comment statement (optional): * Books from VALD and WETH
2	Type SORT after the comment statement
3	Leave at least one blank and type FROM(BKS) BKS specifies the ddname for the input data sets you want to sort.

Table 43 (Page 2 of 2). Steps to Create the SORT Operator

Step	Action
4	<p>Leave at least one blank and type TO(DAPUBS,PRPUBS)</p> <p>TO specifies the ddnames for the output data sets to contain the sorted subset of records. You can create up to 10 identical output data sets of any type that DFSORT allows (permanent, temporary, DASD, tape, print, etc).</p> <p>In this case, DAPUBS is the ddname chosen for the temporary DASD data set and PRPUBS is the ddname chosen for the print data set. You can use any valid 1-8 character ddnames you like.</p> <p>ICETOOL will automatically use OUTFIL to create both output data sets from a single pass over the input data set.</p>
5	<p>Leave at least one blank and type USING(SPUB)</p> <p>USING specifies the first four characters of the ddname for the data set containing the DFSORT control statements. In this case, the four characters chosen are SPUB, but you can use any four characters you like as long as they are valid for a ddname. The last four characters of the ddname are always CNTL, so in this case the full ddname is SPUBCNTL.</p> <p>For the SORT operator, you must specify a SORT control statement in the DFSORT control statement data set (SPUBCNTL) in order to tell DFSORT how to sort the input data set. You can also specify additional DFSORT control statements, like INCLUDE, OMIT, INREC, OUTREC and OUTFIL, as appropriate.</p>

When complete, the SORT operator statement looks like:

```

SORT FROM(BKS) TO(DAPUBS,PRPUBS) USING(SPUB)

```

Diagram illustrating the mapping of ddnames in the SORT operator statement:

- BKS**: ddname of input data set
- DAPUBS**: ddname of first output data set
- PRPUBS**: ddname of second output data set
- SPUB**: First four chars of ddname of DFSORT control data set

To write the JCL statements that go with the SORT operator:

Table 44 (Page 1 of 2). Steps to Create JCL Statements for the SORT Operator

Step	Action
1	<p>Write DD statements for the input data sets and place them at the end of the job. In order to "concatenate" the two input data sets together, you must leave the ddname field blank in the second DD statement:</p> <pre>//BKS DD DSN=A123456.SORT.SAMPIN,DISP=SHR // DD DSN=A123456.SORT.SAMPADD,DISP=SHR</pre>
2	<p>Write DD statements for the DASD and print output data sets and place them at the end of the job:</p> <pre>//DAPUBS DD DSN=&&DSRT,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=SYSDA //PRPUBS DD SYSOUT=A</pre>
3	<p>Write a DD statement for the DFSORT control statement data set and place it at the end of the job:</p> <pre>//SPUBCNTL DD *</pre>

Table 44 (Page 2 of 2). Steps to Create JCL Statements for the SORT Operator

Step	Action
4	<p>Write the SORT control statement to sort the input data sets by publisher and title, and the INCLUDE statement to select only the books by publishers VALD and WETH, and place them after the SPUBCNTL statement:</p> <pre> SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'), FORMAT=CH </pre>

When complete, the TOOLIN statements and DD statements for this SORT operator look like:

```

* Books from VALD and WETH
SORT FROM(BKS) TO(DAPUBS,PRPUBS) USING(SPUB)
/*
//BKS      DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//          DD DSN=A123456.SORT.SAMPADD,DISP=SHR
//DAPUBS   DD DSN=&&DSRT,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=SYSDA
//PRPUBS   DD SYSOUT=A
//SPUBCNTL DD *
            SORT  FIELDS=(106,4,A,1,75,A),FORMAT=CH
            INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),
                        FORMAT=CH
/*

```

Figure 5 on page 82 shows the Book Title and Publisher fields for the records as they would appear in the resulting output data sets. The actual records contain all of the fields.

Book Title	Publisher		
1	75	106	109
CELLS AND HOW THEY WORK		VALD	
COMPLETE SPANISH DICTIONARY		VALD	
EDITING SOFTWARE MANUALS		VALD	
FREUD'S THEORIES		VALD	
INTRODUCTION TO BIOLOGY		VALD	
NOVEL IDEAS		VALD	
SHORT STORIES AND TALL TALES		VALD	
STRATEGIC MARKETING		VALD	
VIDEO GAME DESIGN		VALD	
ZEN BUSINESS		VALD	
ANTICIPATING THE MARKET		WETH	
CIVILIZATION SINCE ROME FELL		WETH	
COMPUTERS: AN INTRODUCTION		WETH	
EIGHTEENTH CENTURY EUROPE		WETH	
GUIDE TO COLLEGE LIFE		WETH	
GUNTHER'S GERMAN DICTIONARY		WETH	
REBIRTH FROM ITALY		WETH	
SYSTEM PROGRAMMING		WETH	
THE INDUSTRIAL REVOLUTION		WETH	

Figure 5. Books from publishers VALD and WETH

Creating Different Subsets of a Sorted Data Set

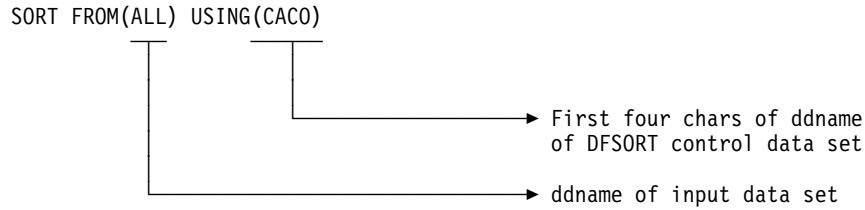
If you want to create subsets of records from the same input data set, you can use OUTFIL statements with a SORT or COPY operator. The OUTFIL statements specify the ddnames of the output data sets, so the TO operand is not needed. All of the features of OUTFIL are available through the SORT and COPY operators.

To write a SORT operator that creates separate DASD and tape data sets for the branch offices in California, and those in Colorado, sorted by city:

Table 45. Steps to Create the SORT Operator

Step	Action
1	Write a comment statement (optional): * Separate output for California and Colorado branches
2	Type SORT after the comment statement
3	Leave at least one blank and type FROM(ALL) ALL specifies the ddname for the input data set you want to sort. You can use the same ddname that you used for A123456.SORT.BRANCH in the STATS operator.
4	Leave at least one blank and type USING(CACO) The CACOCNTL data set contains the SORT and OUTFIL statements.

When complete, the SORT operator statement looks like:



To write the JCL statements that go with the SORT operator:

Table 46. Steps to Create JCL Statements for the SORT Operator

Step	Action
1	Write a DD statement for the DFSORT control statement data set and place it at the end of the job: //CACOCNTL DD *
2	Write the SORT control statement to sort the input data set by city, the OUTFIL statement to select only the California branches, and the OUTFIL statement to select only the Colorado branches, and place them after the CACOCNTL statement: SORT FIELDS=(1,15,CH,A) OUTFIL FNames=(CADASD,CATAPE),INCLUDE=(16,2,CH,EQ,C'CA') OUTFIL FNames=(CODASD,COTAPE),INCLUDE=(16,2,CH,EQ,C'CO')
3	Write DD statements for the DASD and tape output data sets and place them at the end of the job: //CADASD DD DSN=&&CA,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=3390 //CATAPE DD DSN=CA.BRANCH,UNIT=3480,VOL=SER=111111, // DISP=(NEW,KEEP),LABEL=(,SL) //CODASD DD DSN=&&CO,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=3390 //COTAPE DD DSN=CO.BRANCH,UNIT=3480,VOL=SER=222222, // DISP=(NEW,KEEP),LABEL=(,SL)

When complete, the TOOLIN statements and DD statements for this SORT operator look like:

```
* Separate output for California and Colorado branches
SORT FROM(ALL) USING(CACO)
/*
//CACOCNTL DD *
  SORT FIELDS=(1,15,CH,A)
  OUTFIL FNames=(CADASD,CATAPE),INCLUDE=(16,2,CH,EQ,C'CA')
  OUTFIL FNames=(CODASD,COTAPE),INCLUDE=(16,2,CH,EQ,C'CO')
/*
//CADASD DD DSN=&&CA,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=3390
//CATAPE DD DSN=CA.BRANCH,UNIT=3480,VOL=SER=111111,
// DISP=(NEW,KEEP),LABEL=(,SL)
//CODASD DD DSN=&&CO,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=3390
//COTAPE DD DSN=CO.BRANCH,UNIT=3480,VOL=SER=222222,
// DISP=(NEW,KEEP),LABEL=(,SL)
```

Table 47 on page 84 shows the records as they would appear in the CADASD data set (&&CA) and the CATAPE data set (CA.BRANCH) as a result of using the first OUTFIL statement.

Table 47. Records for California Sorted by City

City	State	Employees	Revenue	Profit
1 15	16 17	18 21	22 27	28 33
Los Angeles	CA	32	22530	-4278
Morgan Hill	CA	15	18200	3271
Sacramento	CA	29	42726	8276
San Diego	CA	22	32940	8275
San Francisco	CA	35	42820	6832
San Jose	CA	21	27225	8264
Sunnyvale	CA	18	16152	-978

Table 48 shows the records as they would appear in the CODASD data set (&&CO) and the COTAPE data set (CO.BRANCH) as a result of using the second OUTFIL statement.

Table 48. Records for Colorado Sorted by City

City	State	Employees	Revenue	Profit
1 15	16 17	18 21	22 27	28 33
Aspen	CO	20	25800	5200
Boulder	CO	32	33866	7351
Denver	CO	33	31876	6288
Fort Collins	CO	22	12300	-2863
Vail	CO	19	23202	5027

Figure 8 on page 101 shows the complete TOOLMSG output.

So Far

So far in this tutorial, you have learned how to print statistics for numeric fields using the ICETOOL STATS operator, and how to sort an input data set and create multiple output data sets using the ICETOOL SORT operator. Next, you will learn about the COPY operator.

Creating Multiple Unsorted Data Sets

If you want to create unsorted copies of an input data set, you can use the COPY operator. The COPY operator does not require any DFSORT statements. However, you can supply DFSORT statements (for example, INCLUDE, OMIT, INREC, OUTREC, or OUTFIL) if appropriate.

Here are a couple of examples of COPY operator statements with their accompanying JCL statements:


```
//TOOLIN DD *
  COPY FROM(ALL) TO(D1,D2,D3)
  COPY FROM(ALL) TO(P1) USING(COPY) /*
//ALL DD DSN=A123456.SORT.BRANCH,DISP=SHR
//D1 DD DSN=A123456.SORT.COPY1,DISP=OLD
//D2 DD DSN=A123456.SORT.COPY2,DISP=OLD
//D3 DD DSN=A123456.SORT.COPY3,DISP=OLD
//P1 DD SYSOUT=*
//COPYCNTL DD *
  INCLUDE COND=(16,2,CH,EQ,C'CA') /*
```

The first COPY operator creates identical copies of A123456.SORT.BRANCH in A123456.SORT.COPY1, A123456.SORT.COPY2, and A123456.SORT.COPY3.

The second COPY operator prints the A123456.SORT.BRANCH records for the branches in California. Note that only the character fields in the resulting printed output will be readable. You will learn how to display numeric fields in readable format later in this chapter.

Since copying is more efficient than sorting, you should use the COPY operator rather than the SORT operator when possible.

So Far

So far in this chapter you have learned about STATS, SORT, and COPY, three important ICETOOL operators. The next tutorial shows you how to use the RANGE operator.

Counting Values in a Range

You can use ICETOOL's RANGE operator to count the number of values for a particular numeric field that fall within a range you define. The range can be defined with:

Operand	Comparison
EQUAL	Equal to a value
NOTEQUAL	Not equal to a value
LOWER	Less than a value
HIGHER	Greater than a value
HIGHER and LOWER	Greater than a value, but less than another value

To print a count of the number of California branches with profit greater than -1500, but less than +8000, write the following RANGE statement:

* California branches profit analysis

```
RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
```

Upper limit for range
Lower limit for range
Profit
ddname of input data set

The input data set defined by CADASD is the same data set you created earlier (with the SORT operator) for the California branches. HIGHER(-1500) indicates that you want to count values in the profit field that are greater than -1500, while LOWER(+8000) indicates that you want to count values in the profit field that are less than +8000. For a negative limit, you must specify a minus (-) sign before the number. For a positive limit, you either specify a plus (+) sign before the number or leave it out, therefore, HIGHER(8000) is the same as HIGHER(+8000).

To print a count of the number of branches (in California and Colorado) with less than 32 employees, write the following RANGE statement:

* Branches with less than 32 employees

```
RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
```

Upper limit for range
Employees
ddname of input data set

Since CADASD and ALL were previously defined, you don't need to add any new JCL statements to the ICETOOL job.

When these RANGE operators are run, the results are placed in the TOOLMSG data set. The TOOLMSG output produced for these RANGE operators would look like:

```
* California branches profit analysis
RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
ICE627I 0 DFSORT CALL 0004 FOR COPY FROM CADASD TO E35 EXIT COMPLETED
ICE628I 0 RECORD COUNT: 0000000000000007
ICE631I 0 NUMBER OF VALUES IN RANGE FOR (28,6,PD) : 0000000000000003
ICE602I 0 OPERATION RETURN CODE: 00

* Branches with less than 32 employees
RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
ICE627I 0 DFSORT CALL 0005 FOR COPY FROM ALL TO E35 EXIT COMPLETED
ICE628I 0 RECORD COUNT: 0000000000000012
ICE631I 0 NUMBER OF VALUES IN RANGE FOR (18,4,ZD) : 0000000000000008
ICE602I 0 OPERATION RETURN CODE: 00
```

Looking at the output, you will notice that:

- Message ICE628I gives the count of records processed.

- Message ICE631I gives the count of values in the specified range. There were 3 California branches with profit greater than -1500, but less than +8000, and 8 branches in all with less than 32 employees.
- A return code for each operator is given in message ICE602I.

So Far

You have now learned how to count the number of values in a range for a particular field using the ICETOOL RANGE operator. Next, you will learn about the DISPLAY operator.

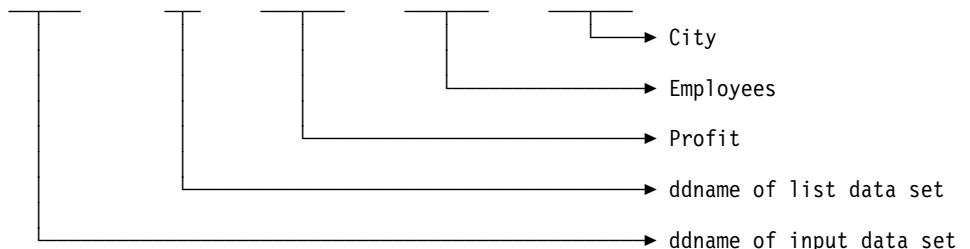
Printing Simple Reports

Numeric fields are often in a format (binary, fixed-point, or decimal) that is not readable when printed. You can use ICETOOL's DISPLAY operator to print simple reports showing numeric and character fields from an input data set in readable form. The specified fields are printed in a list data set that you define. For numeric fields, appropriate plus (+) and minus (-) signs are printed along with the decimal value of each number.

To print a list data set showing the profit, employees, and city fields for the Colorado branches, write the following DISPLAY operator:

* Print profit, employees, and city for each Colorado branch

```
DISPLAY FROM(CODASD) LIST(OUT) ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
```



The input data set defined by CODASD is the same data set you created earlier (with the SORT operator) for the Colorado branches. LIST specifies the ddname for the list data set you want the fields to be printed in. In this case, OUT is the ddname chosen for the list data set, but you can use any valid 1-8 character ddname you like. Specify the ON fields in the same order in which you want their values to be printed in the list data set.

Since OUT has not been defined previously, you must add a JCL statement for it to the end of the job:

```
//OUT DD SYSOUT=A
```

When this DISPLAY operator is run, the OUT data set that results looks like:

(28,6,PD)	(18,4,ZD)	(1,15,CH)
+0000000000005200	+0000000000000020	Aspen
+0000000000007351	+0000000000000032	Boulder
+0000000000006288	+0000000000000033	Denver
-0000000000002863	+0000000000000022	Fort Collins
+0000000000005027	+0000000000000019	Vail

The values for profit, employees, and city are printed in separate columns across the page with a header for each column at the top. If more than one page is printed, DISPLAY puts the header at the top of each page. If you do not want the header printed, you can use DISPLAY's NOHEADER operand to suppress it.

DISPLAY also has two special ON fields you can use:

- **ON(VLEN)** can be used for variable length record data sets to print the length of each record.
- **ON(NUM)** can be used to print a relative record number for each record (starting with 1).

Use the ON(VLEN) or ON(NUM) field just as you would any other ON field.

Printing Tailored Reports

The previous tutorial showed you how to print a simple listing of numeric and character fields using the DISPLAY operator. By using additional operands of DISPLAY, you can create list data sets showing character and numeric fields in a variety of tailored report formats. You can specify:

- Title elements (TITLE, DATE, PAGE, and TIME operands)
- Field headings (HEADER operand)
- Field formats (BLANK and PLUS operands, and formatting items)
- Statistics (TOTAL, AVERAGE, MAXIMUM and MINIMUM operands)
- Lines per page (LINES operand)

To print a report for the Colorado branches showing the city, profit and employee fields with a title line, field headings, totals, averages and minimums, write the following DISPLAY operator:

```

* Print a report for the Colorado branches
DISPLAY FROM(CODASD) LIST(RPT) -
    _____→ ddnames of data sets

DATE TITLE('Colorado Branches Report') PAGE -
    _____→ Title line elements

HEADER('City') HEADER('Profit') HEADER('Employees') -
    _____→ Field headings

ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK -
    _____→ Alternate print format
    _____→ Fields

TOTAL('Total') AVERAGE('Average') MINIMUM('Lowest')
    _____→ Statistics

```

CODASD is the ddname for the previously created Colorado branches data set. RPT is the ddname for the list data set in which you want the report to be printed.

DATE, TITLE and PAGE indicate the elements to be included in the title line and their placement. Specify these operands in the same order in which you want them to be printed in the list data set.

Each HEADER indicates a heading to be used for the corresponding field. Specify the ON fields and their corresponding HEADER strings in the same order in which you want their values to be printed in the list data set. BLANK specifies that numeric values are printed with blank for plus sign, - for minus sign and no leading zeros. HEADER and BLANK also change the justification and column width for the fields to produce a more report-like format.

TOTAL, AVERAGE and MINIMUM cause the indicated statistics to be produced at the end of the report, identified by the specified strings.

Since RPT has not been defined previously, you must add a JCL statement for it at the end of the job:

```
//RPT DD SYSOUT=A
```

When this DISPLAY operator is run, the RPT data set that results looks like:

City	Profit	Employees
-----	-----	-----
Aspen	5200	20
Boulder	7351	32
Denver	6288	33
Fort Collins	-2863	22
Vail	5027	19
Total	21003	126
Average	4200	25
Lowest	-2863	19

The title line and heading line are printed at the top of each page. The character data is left-justified and the numeric data is right-justified with zeros suppressed. The statistics are printed after the columns of data.

Using Formatting Items

The previous tutorial used the BLANK operand to change the way all numeric values in the report are displayed. You can use formatting items to change the appearance of individual numeric fields and their related statistics in the report, with respect to separators, decimal point, decimal places, signs, division, and leading, floating and trailing strings. Formatting items are written as part of the ON operand, separated by commas, as follows: ON(p,m,f,formatting).

Edit Masks

You can select from thirty-three pre-defined edit masks. The table below describes the available masks and shows how the values 12345678 and -1234567 would be printed for each mask. In the pattern:

- **d** is used to represent a decimal digit (0-9)
- **w** is used to represent a leading sign that will be blank for a positive value or - for a negative value
- **x** is used to represent a trailing sign that will be blank for a positive value or - for a negative value
- **y** is used to represent a leading sign that will be blank for a positive value or (for a negative value
- **z** is used to represent a trailing sign that will be blank for a positive value or) for a negative value

Table 49 (Page 1 of 2). Edit Mask Patterns

Mask	Pattern	12345678	-1234567
A0	wdddddddddddddd	12345678	-1234567
A1	wddd,ddd,ddd,ddd,ddd	12,345,678	-1,234,567
A2	wddd.ddd.ddd.ddd.ddd	12.345.678	-1.234.567
A3	wddd ddd ddd ddd ddd	12 345 678	-1 234 567

Table 49 (Page 2 of 2). Edit Mask Patterns

Mask	Pattern	12345678	-1234567
A4	wddd'ddd'ddd'ddd'ddd	12'345'678	-1'234'567
A5	ddd ddd ddd ddd dddx	12 345 678	1 234 567-
B1	wdd,ddd,ddd,ddd,ddd.d	1,234,567.8	-123,456.7
B2	wdd.ddd.ddd.ddd.ddd,d	1.234.567,8	-123.456,7
B3	wdd ddd ddd ddd ddd,d	1 234 567,8	-123 456,7
B4	wdd'ddd'ddd'ddd'ddd.d	1'234'567.8	-123'456.7
B5	wdd'ddd'ddd'ddd'ddd,d	1'234'567,8	-123'456,7
B6	dd ddd ddd ddd ddd,dx	1 234 567,8	123 456,7-
C1	wd,ddd,ddd,ddd,ddd.dd	123,456.78	-12,345.67
C2	wd.ddd.ddd.ddd.ddd,dd	123.456,78	-12.345,67
C3	wd ddd ddd ddd ddd,dd	123 456,78	-12 345,67
C4	wd'ddd'ddd'ddd'ddd.dd	123'456.78	-12'345.67
C5	wd'ddd'ddd'ddd'ddd,dd	123'456,78	-12'345,67
C6	d ddd ddd ddd ddd,ddx	123 456,78	12 345,67-
D1	wddd,ddd,ddd,ddd,ddd	12,345.678	-1,234.567
D2	wddd.ddd.ddd.ddd,ddd	12.345,678	-1.234,567
D3	wddd ddd ddd ddd,ddd	12 345,678	-1 234,567
D4	wddd'ddd'ddd'ddd,ddd	12'345.678	-1'234.567
D5	wddd'ddd'ddd'ddd,ddd	12'345,678	-1'234,567
D6	ddd ddd ddd ddd,dddx	12 345,678	1 234,567-
E1	yddd,ddd,ddd,ddd,dddz	12,345,678	(1,234,567)
E2	yddd.ddd.ddd.ddd,dddz	12.345.678	(1.234.567)
E3	yddd ddd ddd ddd dddz	12 345 678	(1 234 567)
E4	yddd'ddd'ddd'ddd,dddz	12'345'678	(1'234'567)
F1	yd,ddd,ddd,ddd,ddd.ddz	123,456.78	(12,345.67)
F2	yd.ddd.ddd.ddd,ddd,ddz	123.456,78	(12.345,67)
F3	yd ddd ddd ddd ddd,ddz	123 456,78	(12 345,67)
F4	yd'ddd'ddd'ddd,ddd,ddz	123'456.78	(12'345.67)
F5	yd'ddd'ddd'ddd,ddd,ddz	123'456,78	(12'345,67)

To use the E1 edit mask for the Profit field in the previous report, just change ON(28,6,PD) to ON(28,6,PD,E1). The resulting RPT data set then looks like:

10/21/92

Colorado Branches Report

- 1 -

City	Profit	Employees
-----	-----	-----
Aspen	5,200	20
Boulder	7,351	32
Denver	6,288	33
Fort Collins	(2,863)	22
Vail	5,027	19
 Total	 21,003	 126
 Average	 4,200	 25
 Lowest	 (2,863)	 19

Division

You can select from six division items as follows:

- **/K** - divide by 1000
- **/M** - divide by 1000000 (1000*1000)
- **/G** - divide by 1000000000 (1000*1000*1000)
- **/KB** - divide by 1024
- **/MB** - divide by 1048576 (1024*1024)
- **/GB** - divide by 1073741824 (1024*1024*1024)

The Profit values from SORT.BRANCH would look as follows with
HEADER('Profit/(Loss) in K\$') and ON(28,6,PD,E1,**/K**):

```
Profit/(Loss) in K$
-----
          (4)
           6
          (2)
           8
           0
           6
           7
           3
           5
           8
           8
           5
```

Leading, Floating and Trailing Characters

You can add floating characters to your numeric fields and add leading and trailing characters to your numeric and character fields as follows:

- **F'string'** - a floating string to appear to the left of the first non-blank character of the formatted numeric data.
- **L'string'** - a leading string to appear at the beginning of the character or numeric data column.

- **T'string'** - a trailing string to appear at the end of the character or numeric data column.

The Profit values from SORT.BRANCH would look as follows with HEADER('Profit') and ON(28,6,PD,A1,F'\$',T'**'):

```

              Profit
-----
      $-4,278**
       $6,832**
      $-2,863**
       $8,276**
       $-978**
       $6,288**
       $7,351**
       $3,271**
       $5,027**
       $8,264**
       $8,275**
       $5,200**

```

Printing Sectioned Reports

The previous tutorial showed you how to print tailored reports using the DISPLAY operator. By using the BREAK operand of DISPLAY, you can create reports divided into sections by a character or numeric break field on which you have previously sorted. You can also specify a string for the break title (BTITLE operand) and statistics for the individual sections (BTOTAL, BAVERAGE, BMAXIMUM and BMINIMUM operands).

For this example, we will use the data set with books from publishers VALD and WETH, sorted by publisher and title, that we created previously. To print a report with sections by publisher showing the title and price fields with a title line, field headings, break title, break averages and totals, and overall averages and totals, write the following DISPLAY operator:

* Print a report of books for individual publishers

```
DISPLAY FROM(DAPUBS) LIST(SECTIONS) -  
      └───────────────────────────────────┐ ddnames of data sets  
  
TITLE('BOOKS FOR INDIVIDUAL PUBLISHERS') PAGE -  
      └───────────────────────────────────┐ Title line elements  
  
HEADER('TITLE OF BOOK') ON(1,35,CH) -  
      └───────────────────────────────────┐ Heading and field  
  
HEADER('PRICE OF BOOK') ON(170,4,BI,C1,F'$') -  
      └───────────────────────────────────┐ Heading and field  
  
BTITLE('PUBLISHER:') BREAK(106,4,CH) -  
      └──────────┐ └──────────┐  
                  └──────────┐ Break field  
                  └──────────┐ Break title  
  
BAVERAGE('AVERAGE FOR THIS PUBLISHER') -  
      └───────────────────────────────────┐ Section average  
  
BTOTAL('TOTAL FOR THIS PUBLISHER') -  
      └───────────────────────────────────┐ Section total  
  
AVERAGE('AVERAGE FOR ALL PUBLISHERS') -  
      └───────────────────────────────────┐ Overall average  
  
TOTAL('TOTAL FOR ALL PUBLISHERS')  
      └───────────────────────────────────┐ Overall total
```

DAPUBS is the ddname for the previously created VALD and WETH data set. SECTIONS is the ddname for the list data set in which you want the report to be printed.

TITLE and PAGE indicate the elements to be included in the title line and their placement.

Each HEADER and ON pair indicate a field to be included in the report and the heading to be used for it.

BTITLE indicates a string to be used for the break title and its placement (before or after the break field). BREAK indicates the break field to be used to create sections. BAAVERAGE and BTOTAL indicate section statistics to be produced at the end of each section.

AVERAGE and TOTAL indicate overall statistics to be produced at the end of the report.

Since SECTIONS has not been defined previously, you must add a JCL statement for it at the end of the job:

//SECTIONS DD SYSOUT=A

When this DISPLAY operator is run, the SECTIONS data set results in a three-page report that looks as follows:

BOOKS FOR INDIVIDUAL PUBLISHERS - 1 -

PUBLISHER: VALD

TITLE OF BOOK	PRICE OF BOOK
-----	-----
CELLS AND HOW THEY WORK	\$24.95
COMPLETE SPANISH DICTIONARY	\$6.50
EDITING SOFTWARE MANUALS	\$14.50
FREUD'S THEORIES	\$12.50
INTRODUCTION TO BIOLOGY	\$23.50
NOVEL IDEAS	\$24.50
SHORT STORIES AND TALL TALES	\$15.20
STRATEGIC MARKETING	\$23.50
VIDEO GAME DESIGN	\$21.99
ZEN BUSINESS	\$12.00
AVERAGE FOR THIS PUBLISHER	\$17.91
TOTAL FOR THIS PUBLISHER	\$179.14

BOOKS FOR INDIVIDUAL PUBLISHERS - 2 -

PUBLISHER: WETH

TITLE OF BOOK	PRICE OF BOOK
-----	-----
ANTICIPATING THE MARKET	\$20.00
CIVILIZATION SINCE ROME FELL	\$13.50
COMPUTERS: AN INTRODUCTION	\$18.99
EIGHTEENTH CENTURY EUROPE	\$17.90
GUIDE TO COLLEGE LIFE	\$20.00
GUNTHER'S GERMAN DICTIONARY	\$10.88
REBIRTH FROM ITALY	\$25.60
SYSTEM PROGRAMMING	\$31.95
THE INDUSTRIAL REVOLUTION	\$7.95
AVERAGE FOR THIS PUBLISHER	\$18.53
TOTAL FOR THIS PUBLISHER	\$166.77

BOOKS FOR INDIVIDUAL PUBLISHERS - 3 -

TITLE OF BOOK	PRICE OF BOOK
-----	-----
AVERAGE FOR ALL PUBLISHERS	\$18.20
TOTAL FOR ALL PUBLISHERS	\$345.91

So Far

You have now learned how to print simple, tailored and sectioned reports using the ICETOOL DISPLAY operator. Next, you will learn about the OCCUR operator.

Printing How Many Times Fields Occur

You can use ICETOOL's OCCUR operator to print a simple or tailored report showing how many times different field values occur. You can list all of the values in the data set or limit the listing to those values that occur:

- More than once, that is, duplicate values (ALLDUPS operand)
- Only once, that is, non-duplicate values (NODUPS operand)
- A specified number of times (EQUAL operand)
- More than a specified number of times (HIGHER operand)
- Less than a specified number of times (LOWER operand)

For this example, we will use the sample bookstore data set as input. To print a report showing the number of different books in use from each publisher, write the following OCCUR statement:

```
* Print the count of books in use from each publisher
OCCUR    FROM(BKIN) LIST(PUBCT) BLANK -
          └──────────────────────────┘
          │                           │
          │                           └─> Alternate print format
          └──────────────────────────┘
          │                           └─> ddnames of data sets

TITLE('Books from Publishers') DATE(DMY.) -
          └──────────────────────────┘
          │                           └─> Title line elements

HEADER('Publisher') HEADER('Books in Use') -
          └──────────────────────────┘
          │                           └─> Field headings

ON(106,4,CH) ON(VALCNT)
          └──────────────────────────┘
          │                           └─> Publisher and Count
```

BKIN is the ddname for the sample bookstore data set. PUBCT is the ddname for the list data set in which you want the report to be printed. BLANK specifies the field format to be used.

TITLE indicates the title string to appear in the title line. DATE indicates the format in which the date is to appear in the title line (dd.mm.yy where dd is the two-digit day, mm is the two-digit month and yy is the last two digits of the year).

The HEADER strings correspond to the ON fields. ON(VALCNT) is a special ON field used with OCCUR to print the count of occurrences.

Write DD statements for the A123456.SORT.SAMPIN and PUBCT data sets and place them at the end of the job:

```
//BKIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//PUBCT DD SYSOUT=A
```

When this OCCUR operator is run, the PUBCT data set that results looks like:

Books from Publishers 21.10.92

Publisher	Books in Use
COR	7
FERN	4
VALD	5
WETH	4

The name of each publisher is printed along with the number of times that publisher appeared in the sample bookstore data set which is equivalent to the number of different books from that publisher.

So Far

You have now learned how to print a simple or tailored report showing how many times different field values occur. Next, you will learn how to use the SELECT operator.

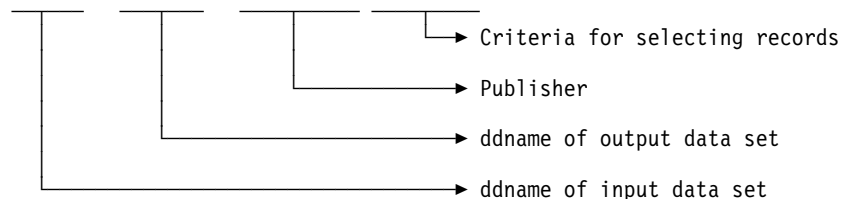
Selecting Records by Field Occurrences

You can use ICETOOL's SELECT operator to create an output data set with records selected according to how many times different field values occur. You can keep only the first record for each value (FIRST operand), only the last record for each value (LAST operand), or only those records with values that occur:

- More than once, that is, duplicate values (ALLDUPS operand)
- Only once, that is, non-duplicate values (NODUPS operand)
- A specified number of times (EQUAL operand)
- More than a specified number of times (HIGHER operand)
- Less than a specified number of times (LOWER operand)

To create an output data set containing records for publishers with more than four different books in use, write the following SELECT statement:

```
* Separate output containing records for publishers
* with more than 4 books in use
SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
```



BKIN is the ddname for the sample bookstore data set. BKOUT is the ddname of the output data set that will contain the records for each publisher field value that occurs more than 4 times (all of the records for COR and VALD in this case).

Write a DD statement for the A123456.BOOKS1 data sets and place it at the end of the job:

```
//BKOUT DD DSN=A123456.BOOKS1,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(3,3)),UNIT=3390
```

Figure 6 shows the Book Title and Publisher fields for the records in the resulting output data set. The actual records contain all of the fields.

Book Title	Publisher		
<hr/>			
1	75	106	109
<hr/>			
LIVING WELL ON A SMALL BUDGET		COR	
SUPPLYING THE DEMAND		COR	
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		COR	
PICK'S POCKET DICTIONARY		COR	
MODERN ANTHOLOGY OF WOMEN POETS		COR	
INTRODUCTION TO PSYCHOLOGY		COR	
CRISES OF THE MIDDLE AGES		COR	
VIDEO GAME DESIGN		VALD	
EDITING SOFTWARE MANUALS		VALD	
STRATEGIC MARKETING		VALD	
SHORT STORIES AND TALL TALES		VALD	
INTRODUCTION TO BIOLOGY		VALD	

Figure 6. Books from Publishers with More than Four Books in Use

So Far

So far in this chapter you have learned how to print statistics for numeric fields, create sorted and unsorted data sets, obtain a count of numeric fields in a range for a particular field, print fields from an input data set, print reports, print a count of field occurrences and select output records based on field occurrences. The last part of this chapter shows the complete ICETOOL job and its resulting TOOLMSG output.

Complete ICETOOL Job and TOOLMSG Output

Here is the complete ICETOOL job you created in this chapter:

```

//EXAMP  JOB  A492,PROGRAMMER
//TOOL   EXEC  PGM=ICETOOL,REGION=1024K
//STEPLIB DD   DSN=A492.SM,DISP=SHR
//TOOLMSG DD   SYSOUT=A
//DFSMSG DD   SYSOUT=A
//TOOLIN DD   *

* Statistics from all branches
STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
* Books from VALD and WETH
SORT FROM(BKS) TO(DAPUBS,PRPUBS) USING(SPUB)
* Separate output for California and Colorado branches
SORT FROM(ALL) USING(CACO)
* California branches profit analysis
  RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
* Branches with less than 32 employees
  RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
* Print profit, employees, and city for each Colorado branch
DISPLAY FROM(CODASD) LIST(OUT) ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
* Print a report for the Colorado branches
DISPLAY FROM(CODASD) LIST(RPT) -
  DATE TITLE('Colorado Branches Report') PAGE -
  HEADER('City') HEADER('Profit') HEADER('Employees') -
  ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK -
  TOTAL('Total') AVERAGE('Average') MINIMUM('Lowest')
* Print a report of books for individual publishers
DISPLAY FROM(DAPUBS) LIST(SECTIONS) -
  TITLE('BOOKS FOR INDIVIDUAL PUBLISHERS') PAGE -
  HEADER('TITLE OF BOOK') ON(1,35,CH) -
  HEADER('PRICE OF BOOK') ON(170,4,BI,C1,F'$') -
  BTITLE('PUBLISHER:') BREAK(106,4,CH) -
  BAVERAGE('AVERAGE FOR THIS PUBLISHER') -
  BTOTAL('TOTAL FOR THIS PUBLISHER') -
  AVERAGE('AVERAGE FOR ALL PUBLISHERS') -
  TOTAL('TOTAL FOR ALL PUBLISHERS')
* Print the count of books in use from each publisher
OCCUR FROM(BKIN) LIST(PUBCT) BLANK -
  TITLE('Books from Publishers') DATE(DMY.) -
  HEADER('Publisher') HEADER('Books in use') -
  ON(106,4,CH) ON(VALCNT)
* Separate output containing records for publishers
* with more than 4 books in use
SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
/*
//ALL DD DSN=A123456.SORT.BRANCH,DISP=SHR
//BKS   DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//      DD DSN=A123456.SORT.SAMPADD,DISP=SHR

```

Figure 7 (Part 1 of 2). Complete ICETOOL Job.

```

//DAPUBS DD DSN=&&DSRT,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=SYSDA
//PRPUBS DD SYSOUT=A
//SPUBCNTL DD *
    SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH
    INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),
        FORMAT=CH
/*
//CACOCNTL DD *
    SORT FIELDS=(1,15,CH,A)
    OUTFIL FNames=(CADASD,CATAPE),INCLUDE=(16,2,CH,EQ,C'CA')
    OUTFIL FNames=(CODASD,COTAPE),INCLUDE=(16,2,CH,EQ,C'CO')
/*
//CADASD DD DSN=&&CA,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=3390
//CATAPE DD DSN=CA.BRANCH,UNIT=3480,VOL=SER=111111,
// DISP=(NEW,KEEP),LABEL=(,SL)
//CODASD DD DSN=&&CO,DISP=(,PASS),SPACE=(CYL,(2,2)),UNIT=3390
//COTAPE DD DSN=CO.BRANCH,UNIT=3480,VOL=SER=222222,
// DISP=(NEW,KEEP),LABEL=(,SL)
//OUT DD SYSOUT=A
//RPT DD SYSOUT=A
//SECTIONS DD SYSOUT=A
//BKIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//PUBCT DD SYSOUT=A
//BKOUT DD DSN=A123456.BOOKS1,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(3,3)),UNIT=3390

```

Figure 7 (Part 2 of 2). Complete ICETOOL Job.

Here is the complete TOOLMSG data set that results from running this job:

```

ICE600I 0 DFSORT ICETOOL UTILITY RUN STARTED

ICE632I 0 SOURCE FOR ICETOOL STATEMENTS:  TOOLIN

ICE630I 0 MODE IN EFFECT:  STOP

      * Statistics from all branches
      STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
ICE627I 0 DFSORT CALL 0001 FOR COPY FROM ALL      TO E35  EXIT COMPLETED
ICE628I 0 RECORD COUNT:  000000000000012
ICE607I 0 STATISTICS FOR (18,4,ZD)      :
ICE608I 0  MINIMUM:  +000000000000015, MAXIMUM:  +000000000000035
ICE609I 0  AVERAGE:  +000000000000024, TOTAL   :  +000000000000298
ICE607I 0 STATISTICS FOR (28,6,PD)      :
ICE608I 0  MINIMUM:  -000000000004278, MAXIMUM:  +000000000008276
ICE609I 0  AVERAGE:  +000000000004222, TOTAL   :  +0000000000050665
ICE607I 0 STATISTICS FOR (22,6,PD)      :
ICE608I 0  MINIMUM:  +0000000000012300, MAXIMUM:  +0000000000042820
ICE609I 0  AVERAGE:  +0000000000027469, TOTAL   :  +00000000000329637
ICE602I 0 OPERATION RETURN CODE:  00

      * Books from VALD and WETH
      SORT FROM(BKS) TO(DAPUBS,PRPUBS) USING(SPUB)
ICE606I 0 DFSORT CALL 0002 FOR SORT FROM BKS      TO OUTFIL   USING SPUBCNTL COMPLETED
ICE602I 0 OPERATION RETURN CODE:  00

      * Separate output for California and Colorado  branches
      SORT FROM(ALL) USING(CACO)
ICE606I 0 DFSORT CALL 0003 FOR SORT FROM ALL      TO OUTFIL   USING CACOCNTL COMPLETED
ICE602I 0 OPERATION RETURN CODE:  00

      * California branches profit analysis
      RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
ICE627I 0 DFSORT CALL 0004 FOR COPY FROM CADASD   TO E35  EXIT COMPLETED
ICE628I 0 RECORD COUNT:  000000000000007
ICE631I 0 NUMBER OF VALUES IN RANGE FOR (28,6,PD)      :  000000000000003
ICE602I 0 OPERATION RETURN CODE:  00

      * Branches with less than 32 employees
      RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
ICE627I 0 DFSORT CALL 0005 FOR COPY FROM ALL      TO E35  EXIT COMPLETED
ICE628I 0 RECORD COUNT:  000000000000012
ICE631I 0 NUMBER OF VALUES IN RANGE FOR (18,4,ZD)      :  000000000000008
ICE602I 0 OPERATION RETURN CODE:  00

      * Print profit, employees, and city for each  Colorado branch
      DISPLAY FROM(CODASD) LIST(OUT) ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
ICE627I 0 DFSORT CALL 0006 FOR COPY FROM CODASD   TO E35  EXIT COMPLETED
ICE603I 0 INFORMATION PRINTED IN OUT      DATA SET
ICE628I 0 RECORD COUNT:  000000000000005
ICE602I 0 OPERATION RETURN CODE:  00

      * Print a report for the Colorado branches
      DISPLAY FROM(CODASD) LIST(RPT) -
      DATE TITLE('Colorado Branches Report') PAGE -
      HEADER('City') HEADER('Profit')  HEADER('Employees') -
      ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK -
      TOTAL('Total') AVERAGE('Average') MINIMUM('Lowest')
ICE627I 0 DFSORT CALL 0007 FOR COPY FROM CODASD   TO E35  EXIT COMPLETED
ICE603I 0 INFORMATION PRINTED IN RPT      DATA SET
ICE628I 0 RECORD COUNT:  000000000000005
ICE602I 0 OPERATION RETURN CODE:  00

```

Figure 8 (Part 1 of 2). Complete TOOLMSG Data Set.

```

      * Print a report of books for individual publishers
      DISPLAY FROM(DAPUBS) LIST(SECTIONS) -
      TITLE('BOOKS FOR INDIVIDUAL PUBLISHERS') PAGE -
      HEADER('TITLE OF BOOK') ON(1,35,CH) -
      HEADER('PRICE OF BOOK') ON(170,4,BI,C1,F'$') -
      BTITLE('PUBLISHER:') BREAK(106,4,CH) -
      BAVERAGE('AVERAGE FOR THIS PUBLISHER') -
      BTOTAL('TOTAL FOR THIS PUBLISHER') -
      AVERAGE('AVERAGE FOR ALL PUBLISHERS') -
      TOTAL('TOTAL FOR ALL PUBLISHERS')
ICE627I 0 DFSORT CALL 0008 FOR COPY FROM DAPUBS TO E35 EXIT COMPLETED
ICE603I 0 INFORMATION PRINTED IN SECTIONS DATA SET
ICE628I 0 RECORD COUNT: 000000000000019
ICE602I 0 OPERATION RETURN CODE: 00

      * Print the count of books in use from each publisher
      OCCUR FROM(BKIN) LIST(PUBCT) BLANK -
      TITLE('Books from Publishers') DATE(DMY.) -
      HEADER('Publisher') HEADER('Books in use') -
      ON(106,4,CH) ON(VALCNT)
ICE627I 0 DFSORT CALL 0009 FOR SORT FROM BKIN TO E35 EXIT COMPLETED
ICE603I 0 INFORMATION PRINTED IN PUBCT DATA SET
ICE628I 0 RECORD COUNT: 000000000000020
ICE638I 0 NUMBER OF RECORDS RESULTING FROM CRITERIA: 000000000000004
ICE602I 0 OPERATION RETURN CODE: 00

      * Separate output containing records for publishers
      * with more than 4 books in use
      SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
ICE627I 0 DFSORT CALL 0010 FOR SORT FROM BKIN TO BKOUT COMPLETED
ICE628I 0 RECORD COUNT: 000000000000020
ICE638I 0 NUMBER OF RECORDS RESULTING FROM CRITERIA: 000000000000012
ICE602I 0 OPERATION RETURN CODE: 00

ICE601I 0 DFSORT ICETOOL UTILITY RUN ENDED - RETURN CODE: 00

```

Figure 8 (Part 2 of 2). Complete TOOLMSG Data Set.

Summary

This chapter covered several of the twelve ICETOOL operators and their uses. For more information on the ICETOOL operators, please refer to *Application Programming Guide*.

Part 4. Learning to Use Symbols

Chapter 11. Defining and Using Symbols	105
Creating the SYMNames Data Set	105
Defining Symbols for Fields	105
Using Symbols for Fields in DFSORT Statements	107
Using Symbols for Fields in ICETOOL Operators	108
Defining and Using Symbols for Constants	109

Chapter 11. Defining and Using Symbols

A *symbol* is a name (preferably something meaningful) you can use to represent a field or a constant. Sets of symbols, also called mappings, can be used to describe a group of related fields and constants such as the information in a particular type of record. Such mappings allow you to refer to fields and constants by their symbols, freeing you from having to know the position, length and format of fields or the values of constants you want to use.

DFSORT's symbol processing feature allows you or your site to create symbols for the fields in your own records and for constants associated with those fields. You can then use those symbols in DFSORT control statements and ICETOOL operators.

In addition, you can obtain predefined sets of symbols to use for the records created by other products such as RACF, DCOLLECT and DFSMSrmm. Visit the DFSORT home page at the following URL to obtain information about downloading these predefined sets of symbols:

<http://www.ibm.com/storage/dfsor/>

The chapter in this section explains how to define symbols for the bookstore data set and use them in several DFSORT control statements and ICETOOL operators. For a full description of how symbols can be used with DFSORT and ICETOOL, see *Application Programming Guide*.

Creating the SYMNAMES Data Set

DFSORT and ICETOOL obtain the symbols to be used from the data set specified in a SYMNAMES DD statement. Create the SYMNAMES data set you want to use with RECFM=FB and LRECL=80 in the same way you would create a data set containing DFSORT JCL and control statements. Then use an editor, such as ISPF EDIT, to write the SYMNAMES statements defining your symbols, as described below.

After you create the SYMNAMES data set, you can use it in any DFSORT or ICETOOL application for which you want to use the symbols you defined. You can add, delete and change SYMNAMES statements in the SYMNAMES data set at any time using the editor.

For this chapter, we will assume you have created a data set called SORT.SYMBOLS to put your SYMNAMES statements in.

Defining Symbols for Fields

Appendix B, "The Sample Bookstore Data Sets" on page 117 shows the fields of the bookstore data set. To define the symbols for these fields, write a SYMNAMES statement for each one in SORT.SYMBOLS. To write SYMNAMES statements that define the symbol *Title* for the Title field and the symbol *Author_Last_Name* for the Author's Last Name field:

Table 50. Steps to Define Symbols for Fields

Step	Action
1	Write a comment statement (optional): * Symbols for fields
2	Type Title followed by a comma. This is the symbol you will use for the Title field. A symbol can be 1 to 50 characters consisting of uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), the number sign (#), the dollar sign (\$), the commercial at sign (@) and the underscore(_). However, the first character must not be a number. Title, TITLE and title are three different symbols.
3	Type the position of the Title field followed by a comma. The position of the Title field is 1 .
4	Type the length of the Title field followed by a comma. The length of the Title field is 75 .
5	Type the format of the Title field followed by a blank. The format of the Title field is CH .
6	Type Author_Last_Name followed by a comma. This is the symbol you will use for the Author's Last Name field.
7	Type * for the position followed by a comma. An asterisk (*) for the position shows that this field immediately follows the previous field. The * here will automatically assign 76 as the position of the Author_Last_Name symbol. You could specify 76 instead of *, but * is preferable when fields are adjacent. * allows symbols for fields to be inserted without changes to symbols for other fields.
8	Type the length of the Author's Last Name field followed by a comma. The length of the Author's Last Name field is 15 .
9	Type the format of the Author's Last Name field followed by a blank. The format of the Author's Last Name field is CH .

When complete, the SYMNames statements for the Title and Author_Last_Name symbols look like:

```
* Symbols for fields
Title,1,75,CH
Author_Last_Name,*,15,CH
```

The SYMNames statements to define the symbols for the other fields in the bookstore data set look like:

```
Author_First_Name,*,15,CH
Publisher,*,4,CH
Course_Department,*,5,CH
Course_Number,*,5,CH
Course_Name,*,25,CH
Instructor_Last_Name,*,15,CH
Instructor_Initials,*,2,CH
Number_in_Stock,*,4,BI
Number_Sold_YTD,*,4,BI
Price,*,4,BI
```

So Far

So far, you have learned how to create a SYMNames data set and use it to define symbols for your fields. Now you will learn how to use the symbols for fields you defined.

Using Symbols for Fields in DFSORT Statements

Now that you have your symbols for the bookstore data set defined in SORT.SYMBOLS, you can use those symbols in DFSORT control statements wherever fields can appear.

Here's an example of a DFSORT application that uses symbols. It selects the books for courses 00032 and 10347 from SORT.SAMPIN and sorts them by title, instructor and price:

```
//SYM1 JOB A492,PROGRAMMER
//SORTIT EXEC PGM=SORT
//STEPLIB DD DSN=A492.SM,DISP=SHR
//SYSOUT DD SYSOUT=A
//SYMNames DD DSN=A123456.SORT.SYMBOLS,DISP=SHR
//SYMNOUT DD SYSOUT=*
//SORTIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//SORTOUT DD DSN=A123456.SORT.SAMPOUT,DISP=SHR
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN DD *
    INCLUDE COND=(Course_Number,EQ,C'00032',OR,
                  Course_Number,EQ,C'10347')
    SORT FIELDS=(Title,A,
                  Instructor_Last_Name,A,Instructor_Initials,A,
                  Price,A)
/*
```

The SYMNames DD statement specifies the SYMNames data set to be used for this application.

The SYMNOUT DD statement specifies a data set in which you want DFSORT to list your original SYMNames statements and the symbol table constructed from them. You can omit the SYMNOUT data set if you don't want to see that information. For this SYMNames data set, the SYMNOUT listing looks like:

----- ORIGINAL STATEMENTS FROM SYMNames -----

* Symbols for fields

Title,1,75,CH

Author_Last_Name,*,15,CH

Author_First_Name,*,15,CH

Publisher,*,4,CH

Course_Department,*,5,CH

Course_Number,*,5,CH

Course_Name,*,25,CH

Instructor_Last_Name,*,15,CH

Instructor_Initials,*,2,CH

Number_in_Stock,*,4,BI

Number_Sold_YTD,*,4,BI

Price,*,4,BI

----- SYMBOL TABLE -----

Title,1,75,CH

Author_Last_Name,76,15,CH

Author_First_Name,91,15,CH

Publisher,106,4,CH

Course_Department,110,5,CH

Course_Number,115,5,CH

Course_Name,120,25,CH

Instructor_Last_Name,145,15,CH

Instructor_Initials,160,2,CH

Number_in_Stock,162,4,BI

Number_Sold_YTD,166,4,BI

Price,170,4,BI

The INCLUDE and SORT statements use symbols for fields instead of position, length and format.

Using Symbols for Fields in ICETOOL Operators

You can also use the symbols from SORT.SYMBOLS in ICETOOL operators and their associated DFSORT control statements wherever fields can appear. Here's an example of an ICETOOL application that uses symbols for fields. It does what the DFSORT example in "Using Symbols for Fields in DFSORT Statements" on page 107 does, but also shows the number of selected books priced below 7 dollars and the number priced above 20 dollars:


```

//SYM2   JOB   A492,PROGRAMMER
//TOOL    EXEC  PGM=ICETOOL,REGION=1024K
//STEPLIB DD DSN=A492.SM,DISP=SHR
//TOOLMSG DD SYSOUT=A
//DFSMSG  DD SYSOUT=A
//SYMNAMES DD DSN=A123456.SORT.SYMBOLS,DISP=SHR
//SYMNOUT DD SYSOUT=*
//IN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//OUT DD DSN=A123456.SORT.SAMPOUT,DISP=SHR
//TOOLIN DD *
      SORT FROM(IN) TO(OUT) USING(CTL1)
      RANGE FROM(OUT) ON(Price) LOWER(700)
      RANGE FROM(OUT) ON(Price) HIGHER(2000)
//CTL1CNTL DD *
      INCLUDE COND=(Course_Number,EQ,C'00032',OR,
                     Course_Number,EQ,C'10347')
      SORT FIELDS=(Title,A,
                     Instructor_Last_Name,A,Instructor_Initials,A,
                     Price,A)
/*

```

The SYMNAMES DD statement specifies the SYMNAMES data set to be used for this application.

The SYMNOUT DD statement specifies a data set in which you want ICETOOL to list your original SYMNAMES statements and the symbol table constructed from them. Since the SYMNAMES data set is the same as for the DFSORT application shown in “Using Symbols for Fields in DFSORT Statements” on page 107, the SYMNOUT listing will be the same as well.

The RANGE operators and the INCLUDE and SORT statements use symbols for fields instead of position, length and format.

So Far

You have now learned how to use symbols for fields in DFSORT control statements and ICETOOL operators. Next, you will learn how to define and use symbols for your constants.

Defining and Using Symbols for Constants

You can use symbols wherever decimal constants, character constants, hexadecimal constants or bit constants can appear in DFSORT control statements and ICETOOL operators.

The ICETOOL example in “Using Symbols for Fields in ICETOOL Operators” on page 108 uses the following RANGE operators:

```

      RANGE FROM(OUT) ON(Price) LOWER(700)
      RANGE FROM(OUT) ON(Price) HIGHER(2000)

```

and the following INCLUDE statement:

```

      INCLUDE COND=(Course_Number,EQ,C'00032',OR,
                     Course_Number,EQ,C'10347')

```

700 and 2000 are decimal constants. C'00032' and C'10347' are character constants. You can use symbols for these constants to make them more understandable in the ICETOOL operators and DFSORT control statements you write. Add symbols for these four constants to the SORT.SYMBOLS data set:

Table 51. Steps to Define Symbols for Constants

Step	Action
1	Write a comment statement (optional): * Symbols for constants
2	Type Discount followed by a comma. This is the symbol you will use for the decimal constant 700.
3	Type the constant followed by a blank. The constant is 700 . You can also use +700 .
4	Type Premium followed by a comma. This is the symbol you will use for the decimal constant 2000.
5	Type the constant followed by a blank. The constant is 2000 . You can also use +2000 .
6	Type Beginning_Economics followed by a comma. This is the symbol you will use for the character constant C'00032'.
7	Type the constant followed by a blank. The constant is C'00032' . You can also use '00032' or c'00032' .
8	Type Advanced_Sociology followed by a comma. This is the symbol you will use for the character constant C'10347'.
9	Type the constant followed by a blank. The constant is C'10347' . You can also use '10347' or c'10347' .

When complete, the SYMNAMEs statements for these constants look like:

```
* Symbols for constants
Discount,700
Premium,2000
Beginning_Economics,C'00032'
Advanced_Sociology,C'10347'
```

The RANGE operators can now be written as:

```
RANGE FROM(OUT) ON(Price) LOWER(Discount)
RANGE FROM(OUT) ON(Price) HIGHER(Premium)
```

The INCLUDE statement can now be written as:

```
INCLUDE COND=(Course_Number,EQ,Beginning_Economics,OR,
              Course_Number,EQ,Advanced_Sociology)
```

Summary

This chapter covered how to define and use symbols for fields and constants in DFSORT control statements and ICETOOL operators. For more information on symbols, please refer to *Application Programming Guide*.

This completes the *Getting Started* tutorials. The appendixes that follow contain valuable information on DFSORT that is also related to this publication.

Part 5. Appendixes

Appendix A. Using the DFSORT Sample Data Sets

The DFSORT product tape includes a sample job, ICEDATA, which creates the data sets SORT.SAMPIN, SORT.SAMPOUT, SORT.SAMPADD, and SORT.BRANCH, to use with the examples in this book. Make sure these data sets are available at your site.

Many of the examples in this book use A123456.SORT.SAMPIN, A123456.SORT.SAMPOUT, A123456.SORT.SAMPADD, and A123456.SORT.BRANCH as the input and output data sets. To match the data set names used in the examples, copy the data sets to your own user ID. Allocate userid.SORT.SAMPIN, userid.SORT.SAMPOUT, and userid.SORT.SAMPADD as fixed-length data sets with a record length of 173 bytes. Allocate userid.SORT.BRANCH as a fixed-length data set with a record length of 33 bytes. Then copy SORT.SAMPIN to userid.SORT.SAMPIN, SORT.SAMPOUT to userid.SORT.SAMPOUT, SORT.SAMPADD to userid.SORT.SAMPADD, and SORT.BRANCH to userid.SORT.BRANCH.

Appendix B. The Sample Bookstore Data Sets

Assume that the sample bookstore data set, SORT.SAMPIN, is used at a college bookstore to keep information about the books it sells. Each horizontal line represents a record, and each column represents a record field. For the sake of illustration, the data set has only 20 records, each 173 bytes long. The data set has also been arranged with headings and numbers to show the byte positions of each field. Neither of these appear in the actual data set. The fields which are in binary format may not appear on your terminal. The methods used to arrange and present that data set as you see it here are explained in the chapters detailing the DFSORT functions.

The first nine fields of each record contain character data and the last three fields contain binary data (binary data is shown in its character representation). Note that because binary data cannot contain decimal points, the prices are shown in cents rather than dollars. Blanks in the course field indicate that the book is not required for any class.

The additional sample data set, SORT.SAMPADD, has 22 records, each 173 bytes long, with the same fields as the sample data set, SORT.SAMPIN.

For your quick reference, the table below shows the length and data format of each field on the sample data sets shown in Appendix C, "Processing Order of Control Statements" on page 121.

Field	Length	Data Format
Title	75	CH
Author's Last Name	15	CH
Author's First Name	15	CH
Publisher	4	CH
Course Department	5	CH
Course Number	5	CH
Course Name	25	CH
Instructor's Last Name	15	CH
Instructor's Initials	2	CH
Number In Stock	4	BI
Number Sold Y-to-D	4	BI
Price	4	BI

Sample Bookstore Data Sets

Sample Data Set - SORT.SAMPIN

Book Title		Author's Last Name		Author's First Name		Publisher		Course Department	
1	75	76	90	91	105	106	109	110	114
COMPUTER LANGUAGES		MURRAY		ROBERT		FERN		COMP	
LIVING WELL ON A SMALL BUDGET		DEWAN		FRANK		COR			
SUPPLYING THE DEMAND		MILLER		TOM		COR		BUSIN	
VIDEO GAME DESIGN		RASMUSSEN		LORI		VALD		COMP	
INKLINGS: AN ANTHOLOGY OF YOUNG POETS		WILDE		KAREN		COR		ENGL	
COMPUTERS: AN INTRODUCTION		DINSHAW		JOKII		WETH		COMP	
PICK'S POCKET DICTIONARY		GUSTLIN		CAROL		COR			
EDITING SOFTWARE MANUALS		OJALVO		VICTOR		VALD		ENGL	
NUMBERING SYSTEMS		BAYLESS		WILLIAM		FERN		COMP	
STRATEGIC MARKETING		YAEGER		MARK		VALD		BUSIN	
THE INDUSTRIAL REVOLUTION		GROSS		DON		WETH		HIST	
MODERN ANTHOLOGY OF WOMEN POETS		COWARD		PETER		COR		ENGL	
INTRODUCTION TO PSYCHOLOGY		DUZET		LINDA		COR		PSYCH	
THE COMPLETE PROOFREADER		GREEN		ANN		FERN		ENGL	
SYSTEM PROGRAMMING		CAUDILLO		RAUL		WETH		COMP	
SHORT STORIES AND TALL TALES		AVRIL		LILIANA		VALD		ENGL	
INTRODUCTION TO BIOLOGY		WU		CHIEN		VALD		BIOL	
ADVANCED TOPICS IN PSYCHOANALYSIS		OSTOICH		DIANNE		FERN		PSYCH	
EIGHTEENTH CENTURY EUROPE		MUNGER		ALICE		WETH		HIST	
CRISES OF THE MIDDLE AGES		BENDER		GREG		COR		HIST	

Sample Data Set - SORT.SAMPADD

Book Title		Author's Last Name		Author's First Name		Publisher		Course Department	
1	75	76	90	91	105	106	109	110	114
GUNTHER'S GERMAN DICTIONARY		WILLIS		GUNTER		WETH			
COMPLETE SPANISH DICTIONARY		ROBERTS		ANGEL		VALD			
ANOTHER ITALIAN DICTIONARY		UNDER		JOAN		COR			
FRENCH TO ENGLISH DICTIONARY		JONES		JACK		FERN			
GUIDE TO COLLEGE LIFE		LAMB		CHARLENE		WETH			
THE ANIMAL KINGDOM		YOUNG		KEVIN		COR			BIOL
A SMALLER WORLD: MICROBES		BEESLY		GEORGE		FERN			BIOL
DNA: BLUEPRINT FOR YOU		IAVERS		ILSE		FERN			BIOL
CELLS AND HOW THEY WORK		JETTS		PETER		VALD			BIOL
KNOW YOUR COMSUMER		ZANE		JENNIFER		COR			BUSIN
ANTICIPATING THE MARKET		ALLEN		CLYDE		WETH			BUSIN
ZEN BUSINESS		WILLIAMS		KATIE		VALD			BUSIN
THE ART OF TAKEOVERS		HUNT		ROBERT		FERN			BUSIN
THE TOY STORE TEST		LITTLE		MARIE		COR			COMP
NOVEL IDEAS		PETERS		SETH		VALD			ENGL
POLITICS AND HISTORY		TOMPSOM		KEN		FERN			HIST
CIVILIZATION SINCE ROME FELL		PIERCE		NICOLE		WETH			HIST
REBIRTH FROM ITALY		FISH		JOHN		WETH			HIST
FREUD'S THEORIES		GOOLE		APRIL		VALD			PSYCH
MAP OF THE HUMAN BRAIN		WINTER		POLLY		COR			PSYCH
QUEUE THEORY		FOX		THAD		FERN			BUSIN
DESIGNING APPLICATIONS		STEVENS		NOAH		COR			COMP

Sample Data Set - SORT.SAMPIN (continued)

Course Number	Course Name	Instructor's Last Name	Instructor's Initials	Number In Stock	Number Sold Year-to-Date	Price
115 119	120 144	145 159	160 161	162 165	166 169	170 173
00032	INTRO TO COMPUTERS	CHATTERJEE	CL	5	29	2600
				14	1	9900
70251	MARKETING	MAXWELL	RF	0	32	1925
00205	VIDEO GAMES	NEUMANN	LB	10	10	2199
10856	MODERN POETRY	FRIEDMAN	KR	2	32	595
00032	INTRO TO COMPUTERS	CHATTERJEE	CL	20	26	1899
				46	38	295
70124	ADVANCED MARKETING	LORCH	MH	3	35	2350
50420	WORLD HISTORY	GOODGOLD	ST	15	9	795
10856	MODERN POETRY	FRIEDMAN	KR	1	26	450
30016	PSYCHOLOGY I	ZABOSKI	RL	26	15	2200
10347	TECHNICAL EDITING	MADRID	MM	7	19	625
00103	DATA MANAGEMENT	SMITH	DC	4	23	3195
10054	FICTION WRITING	BUCK	GR	10	9	1520
80521	BIOLOGY I	GREENBERG	HC	6	11	2350
30975	PSYCHOANALYSIS	NAKATSU	FL	1	12	2600
50632	EUROPEAN HISTORY	BISCARDI	HR	23	21	1790
50521	WORLD HISTORY	WILLERTON	DW	14	17	1200

Sample Data Set - SORT.SAMPADD (continued)

Course Number	Course Name	Instructor's Last Name	Instructor's Initials	Number In Stock	Number Sold Year-to-Date	Price
115 119	120 144	145 159	160 161	162 165	166 169	170 173
				3	16	1088
				8	4	650
				26	6	925
				7	17	1100
				20	1	2000
80522	BIOLOGY II	HAROLD	LM	2	30	3000
80522	BIOLOGY II	HAROLD	LM	9	20	1995
80523	INTRO TO GENETICS	ABRAHAM	NG	15	21	2195
80523	INTRO TO GENETICS	ABRAHAM	NG	8	46	2495
70251	MARKETING	MAXWELL	RF	16	3	4500
70124	ADVANCED MARKETING	LORCH	MH	20	25	2000
70255	BUSINESS THEORY	SCHOFFE	KN	12	12	1200
70255	BUSINESS THEORY	SCHOFFE	KN	17	15	615
00205	VIDEO GAMES	NEUMANN	LB	15	12	2600
10054	FICTION WRITING	BUCK	GR	11	25	2450
50521	WORLD HISTORY	WILLERTON	DW	3	17	995
50420	WORLD HISTORY	GOODGOLD	ST	6	15	1350
50632	EUROPEAN HISTORY	BISCARDI	HR	10	20	2560
30975	PSYCHOANALYSIS	NAKATSU	FL	12	15	1250
30016	PSYCHOLOGY I	ZABOSKI	RL	6	9	895
70255	BUSINESS THEORY	SCHOFFE	KN	2	20	1500
00103	DATA MANAGEMENT	SMITH	DC	7	15	1435

Appendix C. Processing Order of Control Statements

The flowchart below shows the order in which control statements are processed. (SUM is processed at the same time as SORT or MERGE. It is not used with COPY.)

Although you can write the statements in any order, DFSORT always processes the statements in the order shown below.

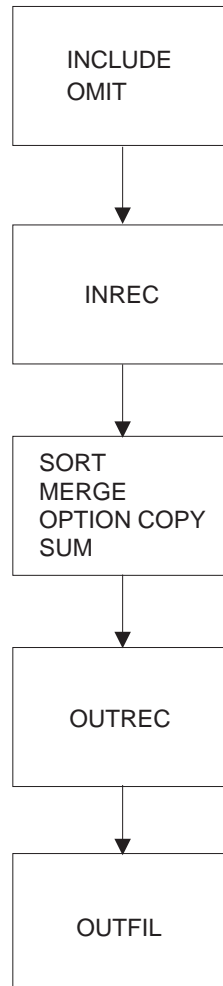


Figure 9. Processing Order of Control Statements

Summary of Changes

Release 13

New Programming Support for Release 13

DFSORT's Performance Booster for The SAS System:** DFSORT Release 13 provides significant CPU time improvements for SAS applications. To take advantage of this new feature, contact SAS Institute Inc. for details of the support they provide to enable this enhancement.

Dynamic Hipersorting: Dynamic Hipersorting is a new, automatic feature that eliminates the unintended system paging activity and expanded storage and paging data set space shortages that sometimes resulted from a large amount of Hipersorting activity, especially from multiple concurrent Hipersorting applications.

Dynamic Hipersorting allows for more optimal DFSORT and system performance and provides installation options that allow you to customize HIPRMAX=OPTIMAL to your own criteria. With the advent of this feature, we recommend that you use HIPRMAX=OPTIMAL as your site default.

Performance: Performance enhancements for DFSORT applications that use the Blockset technique include the following:

- Dataspace sorting, introduced in R12 for fixed-length record sort applications, now available for variable-length record sort applications (MVS/ESA only)
- Improved data processing methods for fixed-length record sort applications
- OUTFIL processing for producing multiple output data sets using a single pass over one or more input data sets.

OUTFIL Processing: OUTFIL is a new DFSORT control statement that allows you to create one or more output data sets for a sort, copy, or merge application from a single pass over one or more input data sets. You can use multiple OUTFIL statements, with each statement specifying the OUTFIL processing to be performed for one or more output data sets. OUTFIL processing begins after all other processing ends (that is, after processing for exits, options, and

other control statements). OUTFIL statements support a wide variety of output data set tasks, including:

- Creation of multiple output data sets containing unedited or edited records from a single pass over one or more input data sets.
- Creation of multiple output data sets containing different ranges or subsets of records from a single pass over one or more input data sets. In addition, records that are not selected for any subset can be saved in another output data set.
- Conversion of variable-length record data sets to fixed-length record data sets.
- Sophisticated editing capabilities such as hexadecimal display and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, symbols (for example, the thousands separator and decimal point), leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.
- Selection of a character or hexadecimal string for output from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).
- Highly detailed three-level (report, page, and section) reports containing a variety of report elements you can specify (for example, current date, current time, page number, character strings, and blank lines) or derive from the input records (for example, character fields, edited numeric input fields, record counts, and edited totals, maximums, minimums, and averages for numeric input fields).

National Language Support

Cultural Sort and Merge: DFSORT will allow the selection of an active locale at installation or run time and will produce sorted or merged records for output according to the collating rules defined in the active locale. This provides sorting and merging for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

Cultural Include and Omit: DFSORT will allow the selection of an active locale at installation or run time and will include or omit records for output according to the collating rules defined in the active locale. This provides inclusion or omission for single- or multi-byte

character data based on defined collating rules which retain the cultural and local characteristics of a language.

OUTFIL Reports: OUTFIL allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world.

ICETOOL Reports: ICETOOL's DISPLAY operator allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world.

ICETOOL Enhancements: ICETOOL is now even more versatile as a result of enhancements to the existing operators. The improvements to ICETOOL include:

- Allowing more data to be displayed with the DISPLAY and OCCUR operators. DISPLAY now allows up to 20 fields (increased from 10) and a line length of up to 2048 characters (increased from 121). OCCUR now allows a line length of up to 2048 characters (increased from 121).
- More extensive formatting capabilities for numeric fields with the DISPLAY operator. Formatting items can be used to change the appearance of individual numeric fields in reports with respect to separators, decimal point, decimal places, signs, division, leading strings, floating strings and trailing strings. Thirty-three pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. Leading and trailing strings can also be used with character fields.
- Display of the four-digit or two-digit year with the DISPLAY and OCCUR operators.
- Division of reports into sections with the DISPLAY operator, based on the values in a character or numeric break field. Statistics (total, maximum, minimum and/or average) can be displayed for each section as well as for the entire report.
- Automatic use of OUTFIL processing for a list of TO ddnames with the COPY and SORT operators, resulting in creation of multiple TO (output) data sets from a single pass over the FROM (input) data set.
- Allowing OUTFIL statements to be specified in the USING data set in addition to or instead of the TO operand with the COPY and SORT operators.
- Allowing the active locale to be specified for the COPY, COUNT and SORT operators, in order to override the installation default for the active locale. Thus, multiple active locales can be used in the same ICETOOL job step for these operators.
- Allowing the last record for each unique field value to be kept with the SELECT operator.

INCLUDE/OMIT Substring Search:

INCLUDE and OMIT function enhancements provide powerful substring search capability to allow inclusion or omission of records when:

- A specified character or hexadecimal constant is found anywhere within a specified input field (that is, a constant is a substring within a field) or
- A specified input value is found anywhere within a specified character or hexadecimal constant (that is, a field is a substring within a constant).

SMF Type-16 Record Enhancements:

New fields, such as information pertaining to each DFSORT run about SORTIN, SORTINnn, SORTOUT and OUTFIL data sets, control statements, record counts, specified values for E15, E35, HIPRMAX, DSPSIZE, FILSZ, LOCALE and AVGRLEN, have been added to DFSORT's SMF type-16 record.

SMF=FULL, SMF=SHORT, and SMF=NO can now be specified in an OPTION statement in DFSPARM or the extended parameter list, to produce or suppress the SMF type-16 record for an individual application.

Note: The offsets of fields ICESPGN, ICEUSER, and ICEGROUP have changed in the Release 13 SMF record. If you have programs that reference those fields, recompile them using the Release 13 version of the ICESMF macro, before attempting to run them against Release 13 SMF records.

Other Enhancements: Several ICEMAC installation options have been added or changed:

- The IBM-supplied default for EXCPVR has been changed from ALL to NONE.
- The IBM-supplied default for DYNAUTO has been changed from NO to YES.
- SDBMSG enables you to specify whether DFSORT should use the system-determined optimum block size for DFSORT message data sets and ICETOOL message and list data sets.
- LOCALE enables you to select an active locale.
- ODMAXBF enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.
- EXPMAX enables you to specify the maximum total amount of available storage to be used for all Hipersorting applications.
- EXPOLD enables you to specify the maximum total amount of old expanded storage to be used at any one time by all Hipersorting applications.

- EXPRES enables you to specify the minimum amount of available expanded storage to be reserved by DFSORT for use by non-Hipersorting applications.

Several run-time options have been added or changed:

- LOCALE enables you to select an active locale.
- SMF enables you to specify whether DFSORT is to produce SMF type-16 records.
- ODMAXBF enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.
- NZDPRINT enables you to indicate that positive ZD summation results are not to be converted to printable numbers (overrides ZDPRINT).
- HILEVEL=YES on the MODS statement enables you to indicate that the E15 and E35 routines are to be treated as COBOL exits.
- DEBUG options BUFFERS=ANY and BUFFERS=BELOW will now be recognized but not used.

DFSORT will now ignore any DD statements not needed for the application (for example, a SORTIN DD statement will be ignored for a merge application).

For unsuccessful completion due to an unsupported operating system, DFSORT, ICEGENER, and ICETOOL will now pass back a return code of 24 to the operating system or invoking program.

The installation initialization exit, ICEIEXIT, enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.

The installation termination exit, ICETEXIT, contains additional fields such as a flag to indicate that OUTFIL processing was used.

For INREC and OUTREC:

- The upper limit for columns and the end of fields has been raised from 32000 to 32752.
- 1: before the RDW field of variable-length records will be accepted and ignored.

For INCLUDE and OMIT, COND=ALL, COND=(ALL), COND=NONE, and COND=(NONE) enable you to include or omit all records.

The L2 value from the RECORD statement will be used if the L1 value is not specified when an E15 or E32 user exit passes all of the input records.

When input is a VSAM data set and output is a non-VSAM data set with RECFM not specified, DFSORT will now set the output RECFM as blocked rather than unblocked, when doing so will allow the use of the system-determined optimum block size for output.

New Programming Support for Release 12 (PTFs)

ICEGENER, copy, and Blockset sort and merge can now be used when a tape output data set is specified with DISP=MOD or DISP=OLD, without specifying the RECFM, LRECL, or BLKSIZE in the DD statement.

Sequential striping is supported for input and output data sets.

Compression is supported for input and output data sets.

BatchPipes/MVS input and output pipes are supported.

New Device Support for Release 12 (PTFs)

Four-digit device numbers are supported.

The IBM 3390-9 DASD is supported for input, output, and work data sets, although it is not recommended for work data sets for performance reasons.

The IBM RAMAC Array DASD and RAMAC Array Subsystem are supported for input, output, and work data sets.

The IBM 3990 Model 6 control unit is supported.

The IBM cached 9343 control unit models are supported.

Index

A

- allowable comparisons 25
 - ascending order
 - example 3
 - sorting 11
- ASCII data format code 4

B

- BI format 4
- binary data 4
- binary zeros 33
- blanks 33

C

- calling DFSORT
 - COBOL program 57
 - PL/I program 63
- CH format 4
- character data 4
- character strings, format 34
- COBOL
 - calling DFSORT 57
 - MERGE statement 57
 - passing DFSORT statements 57
 - sample program 58, 60
 - SORT statement 57
- COBOL II
 - FASTSORT compile-time option 69
 - passing DFSORT statements 63
 - comparison
 - allowable 25
 - field-to-field 21
 - field-to-constant 21, 25
 - field-to-field 25
 - operators 22
- constant
 - format for writing 26
 - inserting with OUTREC statement 34
- continuing a statement 14
- control fields
 - combining 14
 - equally collating 7
 - multiple 13
 - overlapping 13
 - summing 27
- control statements
 - ordering 22
 - writing 6
- control fields
 - deleting duplicate records 29

- control fields (*continued*)
 - duplicate 27
 - equally collating 27
 - general information 6
 - reordering 32
- control statements
 - processing order flowchart 121
- COPY statement 19
- COPY operator (ICETOOL) 73, 84
- copying records 19
- COUNT operator (ICETOOL) 73
- creating a DFSORT job 5

D

- data format 4
- data set
 - copying
 - definition 5
 - increasing speed 21
 - writing the COPY statement 19
 - making multiple copies 42
 - merging
 - definition 4
 - using program control statements 16
 - sorting
 - ascending order 11
 - by multiple control fields 13
 - definition 3
 - descending order 12
 - increasing speed 21
 - writing the SORT statement 11
- data set
 - merging
 - increasing speed 21
 - tailoring with INCLUDE 21
 - tailoring with OMIT 21
- DD statement
 - SORTIN 15
 - SORTOFn 42
 - SORTOUT 15
 - SORTWKdd 15, 67
 - STEPLIB 15
 - SYSIN 15
 - SYSOUT 15
- DD statement
 - SORTCNTL 57
 - SORTOUT 32
- defaults
 - order of equal records 7
 - overriding 65
 - using OPTION statement 65
 - using PARM parameter 65

- DEFAULTS operator (ICETOOL) 73
- deleting fields 31
- deleting records 27
 - descending order
 - example 3
 - sorting 12
- devices 67
 - DFSPARM 15, 66
- DISPLAY operator (ICETOOL) 73, 87, 88, 90, 93
- DUMMY TAG
 - ISMF profile 63
- duplicate control fields 27
- duplicate fields
 - OCCUR operator (ICETOOL) 73, 96
 - SELECT operator (ICETOOL) 73, 97
 - UNIQUE operator (ICETOOL) 74

E

- EBCDIC data format code 4
- edit masks
 - ICETOOL 90
 - OUTFIL 46
- editing records 31
- equal control fields 7
- excluding records 24
- EXEC statement
 - merging 18
 - sorting 15

F

- FASTSORT
 - COBOL 63
 - compile-time option 69
- field-to-constant comparison 21, 25
- field-to-field comparison 25
 - fields
 - character and numeric, printing 87
 - deleting unnecessary fields 31
 - numeric and character, printing 87
 - printing statistics for numeric 77
 - reformatting 32
- FIELDS=NONE, specifying on SUM statement 29
- FILES parameter 42
- FNAMES parameter 42
- FNAMES parameter 49
 - format
 - BI 4
 - CH 4
 - character string 26
 - character strings 34
 - data 4
 - decimal string 26
 - hexadecimal string 26
 - hexadecimal strings 34

format (*continued*)

- PD 4
- ZD 4
- FORMAT parameter 14
- formats for writing constants 26

H

- HEADER2 parameter 50
- HEADERn parameter 45
 - hexadecimal strings, format 34
- high-speed DASD 67
- Hiperspace 67

I

- ICEDATA (sample job) 8
- ICEDATA (sample job) 115
- ICETOOL
 - complete sample job 98
 - continuing an operator statement 78
 - COPY operator 73, 84
 - COUNT operator 73
 - creating a job 75
 - DEFAULTS operator 73
 - definition 73
 - DISPLAY operator 73, 87, 88, 90, 93
 - examples 88, 90, 93, 96, 97
 - complete ICETOOL job 98
 - continuing an operator statement 78
 - counting values in a range 85
 - creating a job 75
 - creating different subsets 82
 - creating multiple sorted data sets 79
 - creating multiple unsorted data sets 84
 - printing simple reports 87
 - printing statistics for numeric fields 77
 - printing statistics for record lengths 79
 - TOOLMSG output 98
 - writing required JCL 75
 - JCL statements 75
 - job
 - elements 75
 - sample 98
 - MODE operator 73
 - OCCUR operator 73, 96
 - operator summary 73
 - printing
 - fields 87
 - record length 88
 - relative record number 88
 - statistics for numeric fields 77
 - printing field value occurrences 96
 - printing sectioned reports 93
 - printing tailored reports 88
 - RANGE operator 73, 85

- ICETOOL (*continued*)
 - reports 45
 - requirements
 - input data sets 74
 - JCL 75
 - SELECT operator 73, 97
 - selecting records by field occurrences 97
 - SORT operator 73, 79, 82
 - statements
 - blank 76
 - comment 76
 - operator, continuing 78
 - STATS operator 74, 77, 79
 - symbols 108
 - TOOLMSG output, sample 98
 - UNIQUE operator 74
 - using formatting items 90
 - VERIFY operator 74
- improving performance
 - allocating main storage 67
 - FASTSORT compile-time option 69
 - INCLUDE statement 68
 - INREC statement 68
 - JCL (job control language) 69
 - OMIT statement 68
 - options to avoid 69
 - SKIPREC option 68
 - SUM statement 68
- improving performance
 - high-speed DASD 67
 - Hiperspace 67
 - STOPAFT option 68
 - using devices 67
- INCLUDE parameter 43
- INCLUDE statement
 - allowable comparisons 25
 - improving performance 68
 - using with INREC 38
 - writing 21
- increasing speed
 - copying 21
 - merging 21
 - sorting 21
- input data set
 - selecting 15
 - tailoring with OMIT 21
 - tailoring with INCLUDE 21
- input data set
 - ICETOOL 74
- INREC statement
 - considerations when used with other statements 38
 - considerations when used with other statements 38
 - deleting fields 31
 - differences between INREC and OUTREC processing 31

- INREC statement (*continued*)
 - improving performance 68
 - inserting separators 31
 - processing order 38
 - reformatting records 37
 - reordering fields 31
- INREC statement
 - processing order flowchart 121
- inserting with OUTREC
 - binary zeros 32
- inserting with OUTREC
 - blanks 33
 - blanks or zeros 31
 - constants 34
- installation defaults, overriding 65
- ISCI data format code 4

J

- JCL (job control language)
 - selecting output data sets 15
- JCL (job control language)
 - executing a copy 20
 - executing a merge 18
 - executing a sort 15
 - general information 15
 - improving performance 69
 - selecting input data sets 15
 - statements 15
- JCL (job control language)
 - calling DFSORT from a program 57, 60, 63
- ICETOOL 75
- JOB statement 15

L

- LINES parameter 45, 49
- logical record length 31

M

- main storage, allocating 67
- making multiple data set copies
 - FILES parameter 42
- making multiple data set copies
 - FNAMES parameter 42
- mapping
 - symbols 105
- margins 33
- MERGE statement
 - specifying with COPY 19
 - writing 17
- MERGE statement, COBOL 57
- merging records 16
- MODE operator (ICETOOL) 73

- multiple control fields 13
- multiple data set copies, making
 - FNAMES parameter 42
- multiple data set copies, making
 - FILES parameter 42

N

- non-duplicate fields
 - OCCUR operator (ICETOOL) 73, 96
 - SELECT operator (ICETOOL) 73, 97
 - UNIQUE operator (ICETOOL) 74
- notices vii

O

- OCCUR operator (ICETOOL) 73, 96
- occurrences
 - OCCUR operator (ICETOOL) 73, 96
 - SELECT operator (ICETOOL) 73, 97
- OMIT parameter 43
- OMIT statement
 - allowable comparisons 25
 - improving performance 68
 - using with INREC 38
 - writing 24
- OPTION COPY 20
- OPTION statement 19
- OPTION statement 65
- order of control statements 22
 - ordering
 - ascending order 3
 - control statements 22
 - descending order 3
 - DFSORT defaults 7
- OUTFIL
 - edit mask patterns 46
 - reports 45
- OUTFIL reports
 - sample 54
 - using FNAMES parameter 49
 - using HEADER2 parameter 50
 - using LINES parameter 49
 - using OUTREC parameter 52
 - using TRAILER1 parameter 54
- OUTFIL statement
 - creating a report 45
 - making multiple data set copies 42
 - making multiple output data sets with unique content 43
 - parameters
 - FNAMES 42
 - HEADER1 45
 - HEADER2 45
 - INCLUDE 43
 - LINES 45
 - OMIT 43

- OUTFIL statement (*continued*)
 - parameters (*continued*)
 - OUTREC 45
 - TRAILER 1 45
 - TRAILER2 45
- OUTFIL statement
 - making multiple data set copies 42
 - parameters
 - FILES 42
 - processing order flowchart 121
 - using 41, 56
- output data set 15
 - output records, reformatting 31
- OUTREC parameter 45, 52
- OUTREC statement
 - deleting fields 31
 - editing records 31
 - padding 31
 - setting up report format 35
 - using with INREC 38
 - using to
 - insert blanks 33
 - insert constants 34
 - insert binary zeros 33
 - writing 31
- OUTREC statement
 - deleting fields 31
 - differences between INREC and OUTREC processing 31
 - inserting separators 31
 - processing order flowchart 121
 - reordering fields 31
- overflow
 - explanation 29
 - preventing 40
- overriding installation defaults 65

P

- packed decimal data 4
- padding
 - description 23
 - using INCLUDE 23
 - using OMIT 23
 - using OUTREC statement 31
- PARM parameter 65
- passing DFSORT statements
 - from a COBOL II program 63
 - from a COBOL program 57
 - from a PL/I program 57
- PD format 4
- PL/I
 - calling DFSORT 63
 - passing DFSORT statements 57
 - sample program 63

- printing reports
 - DISPLAY operator (ICETOOL) 73, 87, 88, 90, 93
- processing order
 - differences between INREC and OUTREC 31
- processing order
 - flowchart 121
 - special considerations for INREC 38
 - special considerations for INREC 38

R

- RANGE operator (ICETOOL) 73, 85
- record length, changing 32
- RECORD statement 63
- records
 - considerations when reordering with INREC and OUTREC 37
 - considerations when reordering with INREC and OUTREC 38
 - copying
 - definition 5
 - writing the COPY statement 19
 - merging
 - definition 4
 - using program control statements 16
 - reformatting with OUTREC 21
 - reordering with OUTREC 21
 - reordering and reformatting with OUTREC 32
 - selecting by occurrences 97
 - sorting
 - ascending order 11
 - by multiple control fields 13
 - definition 3
 - descending order 12
 - writing the SORT statement 11
 - summing 27
- reformatting records 31
- reordering fields 32
- reports 87, 88, 90, 93
 - DISPLAY operator (ICETOOL) 73, 90
 - division items (ICETOOL) 92
 - edit masks 90
 - ICETOOL 87
 - ICETOOL vs OUTFIL 45
 - leading, floating and trailing characters (ICETOOL) 92
 - OUTFIL sample 54
 - OUTREC statement 35
 - printing field value occurrences 96
 - sectioned (ICETOOL) 93
 - simple (ICETOOL) 87
 - tailored (ICETOOL) 88
 - using OUTFIL 45
- return code, DFSORT (COBOL) 58
- running a DFSORT job 5

S

- sample data set 8
- sample data set
 - SORT.SAMPADD 118
 - SORT.SAMPIN 117
- sample data set
 - SORT.BRANCH 115
 - SORT.SAMPIN 115
 - SORT.SAMPOUT 115
 - using 115
- SELECT operator (ICETOOL) 73, 97
- selecting input data sets 15
- selecting output data sets 15
- selecting records 21, 97
 - SKIPREC option 68
- SORT.SAMPADD 8
- SORT statement
 - specifying COPY 19
 - using with INREC 38
 - writing 11
- SORT operator (ICETOOL) 73, 79, 82
- SORT statement, COBOL 57
- SORT-CONTROL special register (COBOL) 58
- SORT-RETURN special register (COBOL) 58
- SORT.BRANCH 8, 115
- SORT.SAMPADD 118
- SORT.SAMPIN 8, 115, 117
- SORT.SAMPOUT 8, 115
- SORTCNTL DD statement 57
- SORTIN DD statement 15
- sorting records
 - ascending order 11
 - descending order 12
 - increasing speed 21
 - writing the SORT statement 11
- sorting records
 - by multiple control fields 13
 - calling DFSORT from a program 57
 - FASTSORT option 63
- SORTINnn DD statement 18
- SORTOFn DD statement 42
- SORTOUT DD statement
 - changing record length 32
 - general information 15
- SORTWKdd DD statement 15, 67
- SORTWKdd DD statement 18
- specifying COPY
 - OPTION statement 19
 - specifying COPY
 - MERGE statement 19
 - SORT statement 19
- statistics, printing for numeric fields 77
- STATS operator (ICETOOL) 74, 77
- STEPLIB DD statement 15

- STOPAFT option 68
- SUM statement
 - FIELDS=NONE 29
 - improving performance 68
 - writing 27
- SUM statement
 - using with INREC 38
- summary fields 27
- symbols
 - constants 109
 - control statement
 - example 107
 - DCOLLECT 105
 - definition 105
 - DFSMSrmm 105
 - fields 105
 - ICETOOL 108
 - RACF 105
 - SYMNAMES statement 105
- SYMNAMES statement
 - symbols 105
- SYSIN DD statement 15
- SYSOUT DD statement 15

T

- tailoring a data set
 - using INCLUDE 21
 - using OMIT 21
- TRAILER1 parameter 54
- TRAILERn parameter 45
- truncation 23

U

- UNIQUE operator (ICETOOL) 74
- using the DFSORT sample data sets 115

V

- VERIFY operator (ICETOOL) 74

W

- work storage data sets
 - devices for efficient use 67
 - work storage data sets
 - number needed 15

Z

- ZD format 4
- zeros 33
- zoned decimal data 4

Readers' Comments — We'd Like to Hear from You

DFSORT

Getting Started with DFSORT
Release 14

Publication No. SC26-4109-08

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



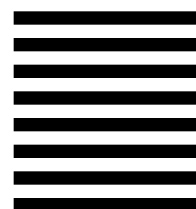
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
RCF Processing Department
G26/M86 050
5600 Cottle Road
SAN JOSE, CA 95193-0001



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5740-SM1



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-4109-08

